Service Entitlement Configuration
Version 13.0
29 January 2026

Security Classification: Non-confidential

Copyright Notice

Disclaimer

Compliance Notice

The information contain herein is in full compliance with the GSMA Antitrust Compliance Policy.

This Permanent Reference Document is classified by GSMA as an Industry Specification, as such it has been developed and is maintained by GSMA in accordance with the provisions set out GSMA AA.35 - Procedures for Industry Specifications.

.

Table of Contents

1 Introduction

1.1 Overview

This document describes the procedure for configuration of a device-based service performed during the entitlement verification step of the service or during the activation of that service.

The device services covered in this document are Voice-over-Wi-Fi (VoWiFi), Voice-over-Cellular (4G VoLTE and 5G VoNR), SMS over IP (SMSoIP) and On-Device Service Activation (ODSA) of Companion devices (associated with a requesting device) and Primary devices.

The specification leverages the protocol and document presentation described in GSMA PRD RCC.14 [5]. In this context, the term “entitlement” refers to the applicability, availability, and status of that service (or feature) on a device.

The entitlement configuration is exchanged between a VoWiFi, Voice-over-Cellular, SMSoIP, Companion ODSA or Primary ODSA client on a device and a Service Provider’s Entitlement Configuration Server. It is independent from the service configuration procedure between clients and the Service Provider’s configuration server described in GSMA PRD RCC.14 [5].

Entitlement configuration defines a mechanism for a Service Provider to inform mobile devices of the status of IP Multimedia Subsystem (IMS) network services like VoWiFi, Voice-over-Cellular and SMSoIP.

In the ODSA context it defines the interaction between an ODSA client, a client application on a device that entitles and activates a companion or primary device’s subscription, and the Service Provider.

This procedure leverages the subscription profile of the end-user, identified by the SIM card, and the network’s readiness in supporting the service. The entitlement client can then dynamically activate (or deactivate) the service according to the activation (respectively deactivation) status retrieved from the Service Provider’s Entitlement Configuration Server.

When required by the service, entitlement configuration also covers on-device service activation flow, for example to display a web page describing the service or to get end-user consent on the service’s Terms and Conditions.

Service configuration in this document deals with the configuration parameters controlling the entitlement of a service. Those parameters come in addition to the ones defined in GSMA PRD IR.51 [2] and GSMA PRD IR.92 [3] that relate to the internal settings and configuration of IMS services. IMS service configuration as defined in GSMA PRD IR.51 [2] and GSMA PRD IR.92 [3] are out of scope.

1.2 In Scope

This document covers both the device and network aspects of the entitlement configuration for VoWiFi, Voice-over-Cellular and SMSoIP services as well as for On-Device Service Activation (ODSA) of Companion and Primary devices. Service-specific aspects need to be

described in documents relating to those services as in GSMA PRDs IR.51 [2] and IR.92 [3] for IMS services.

The entitlement configuration can be obtained via either cellular or Wi-Fi data connectivity. In case Wi-Fi data connection is used, this document assumes that a Wi-Fi bearer is available to the device and the requirements of that Wi-Fi bearer conform to GSMA PRD TS.22 [7]. Configuration and provisioning of the Wi-Fi bearer is described in GSMA PRD TS.22 [7] Section 3.

1.3 Interactions with Other GSMA Specifications

Entitlement configuration is an optional mechanism between applications/services on devices (like VoWiFi and Voice-over-Cellular) and the Service Provider that occurs during service activation. The procedure requires both end-user's subscription data and network readiness information from the SP.

To support that exchange, an entitlement configuration server leverages the GSMA PRD RCC.14 [5] protocol to carry the required entitlement data between devices' applications and the Service Provider. The entitlement configuration procedure is separate from the service configuration procedure specified in GSMA PRD RCC.14 [5]. A device or application shall not query for both entitlement and service configurations in the same request.

The result of entitlement configuration for a service offers the assurance that the end-user's associated subscription and the core network's readiness have been verified, allowing the service to be offered to the end-user.

Note: in the following sub-sections of 1.3, Voice-over-Cellular (VoLTE Only) is used to compare with other GSMA specifications.

1.3.1 Positioning of VoWiFi, VoLTE and SMSoIP entitlements with respect to TAD and MNO Provisioning

The positioning of VoWiFi, VoLTE and SMSoIP entitlement configuration with respect to existing GSMA device configuration procedures (GSMA PRD TS.32 [8], GSMA PRD IR.51 [2] and GSMA PRD IR.92 [3]) is presented in Figure 1. It shows the typical timeline and triggers that would induce the procedures (note that the horizontal axis represents Time).

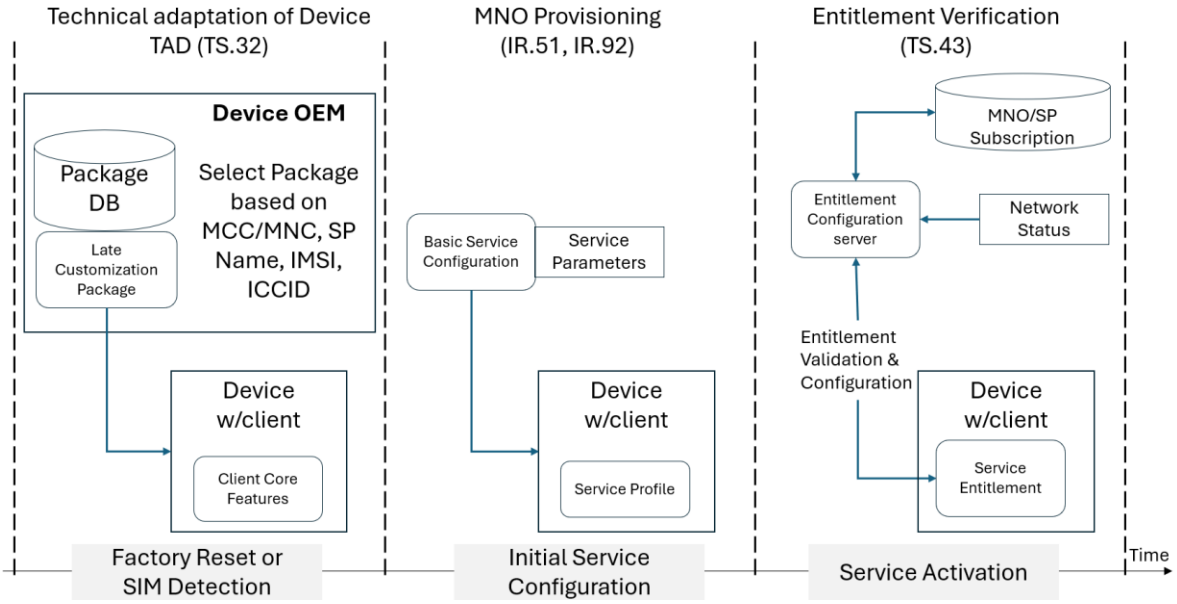

Figure 1. TS.43 VoWiFi, VoLTE and SMSoIP entitlement procedure with respect to TS.32, IR.51 and IR.92

The GSMA PRD TS.32 [8] procedure of Technical Adaptation of Device (TAD) is implemented by device OEMs on an MNO-wide basis (or a range of IMSI) due to the device's factory reset or SIM detection. General IMS, VoLTE and VoWiFi parameter values are set without taking into account end-user subscription or network related information.

The MNO provisioning procedure of GSMA PRD IR.51 [2] and IR.92 [3] also offers the possibility of setting general IMS, VoLTE and VoWiFi parameters on the device during initial service configuration. However, it is not associated with user-triggered service activation or the verification of the services' entitlement / applicability.

The entitlement-level configuration for VoLTE and VoWiFi specified in the GSMA PRD TS.43 takes place after or outside the aforementioned GSMA's device and service configuration procedures. It is also triggered by events not associated with GSMA PRD TS.32 [8], GSMA PRD IR.51 [2] and GSMA PRD IR.92 [3]:

- when the service needs to verify its entitlement status (during service initiation),
- when the end-user wishes to activate the service (via the service's settings menu)

1.3.2 Relationship with TS.32, IR.51 and IR.92 VoWiFi/VoLTE/SMSoIP Parameters

The VoWiFi, VoLTE and SMSoIP configuration parameters of this PRD complement the ones from GSMA PRD TS.32 [8], GSMA PRD IR.51 [2] and GSMA PRD IR.92 [3].

While those specifications define general-purpose VoWiFi, VoLTE and SMSoIP parameters to enable or disable those services on the device, the GSMA PRD TS.43 defines parameters

that relate to service initiation and end-user activation (capture of Terms & Conditions, capture of physical address).

The parameters in this PRD are also based on end-user subscription's data and on the network readiness for those services.

In case the VoWiFi, VoLTE or SMSoIP service has not been allowed and activated on the device due to a Technical Adaptation of Device (TAD) or MNO provisioning procedure, the client performing the entitlement configuration should be disabled.

The VoLTE, SMSoIP and VoWiFi configuration parameters defined in each specification are presented in Table 1Table 1. VoLTE, SMSoIP and VoWiFi Configuration Parameters in GSMA Specifications

.

GSMA PRD	VoLTE Status Parameters	SMSoIP Status Parameters	VoWiFi Status Parameters
GSMA PRD TS.32[8]	• **VxLTE 1.27** Voice/Video over LTE allowed when roaming. • **VxLTE 1.28** Voice/Video over LTE allowed	• **VxLTE 1.07** SMSoIP Networks Indications (not used or preferred)	• **VoWiFi 3.01** Voice and Video / Voice enabled over Wi-Fi
GSMA PRD IR.92 [3]	As a Media_type_restriction_policy • Voice and/or Video over LTE allowed. • Voice and/or Video over LTE allowed while roaming	SMSoIP_usage_policy (When to use SMSoIP)	N/A
GSMA PRD IR.51[2]	N/A	N/A	As a Media_type_restriction_policy • Voice and/or Video over Wi-Fi enabled
TS.43 (this document)	• VoLTE entitlement status	• SMSoIP entitlement status	• VoWiFi entitlement status • VoWiFi T&Cs capture status • VoWiFi address capture status • VoWiFi provisioning status

Table 1. VoLTE, SMSoIP and VoWiFi Configuration Parameters in GSMA Specifications

Note: That the configuration parameter VxLTE 1.21 - IMS Enabled (Yes/No) from TS.32 [8] and "IMS Status" from IR.92 [3] is not impacted by the GSMA PRD TS.43. The overall IMS function on the device can still be controlled by this parameter.

1.3.3 Controlling Access to Network and PS Data for Entitlement Configuration

GSMA PRD IR.92 [3] defines parameters to allow device and client services to be exempt of the 3GPP PS Data Off feature. When one such parameter, **Device_management_over_PS**, is set, it indicates that device management over PS is a 3GPP PS data off exempt service.

GSMA PRD TS.43 extends the **Device_management_over_PS** parameter to include Entitlement Configuration as a type of “device management” service that can be exempt of 3GPP PS Data Off.

The home operator can also configure a policy on the Entitlement Client around the access type used during entitlement configuration. This is done with the **`AccessForEntitlement`** parameter with values listed in Table 2.

AccessForEntitlement Value	Description
0	any access type
1	3GPP accesses only
2	WLAN/Wi-Fi only
3	3GPP accesses preferred, WLAN/Wi-Fi as secondary
4	WLAN/Wi-Fi preferred, 3GPP accesses as secondary
5-255	not assigned

Table 2. AccessForEntitlement Parameter

A "not assigned" value is interpreted as "any access type" value.

When not preconfigured by the home operator with the **`AccessForEntitlement`** parameter, the Entitlement Client shall perform entitlement configuration requests over Wi-Fi if available. When there is no Wi-Fi connectivity, the Entitlement Client shall perform requests over cellular if it is not forbidden (i.e. PS data off and not exempt).

1.4 Abbreviations

Abbreviation	Definition
APNS	Apple Push Notification Service
CP AC	Client Provisioning Application Characteristic
DNS	Domain Name Server
EAP-AKA	Extensible Authentication Protocol for 3rd Generation Authentication and Key Agreement
EID	eUICC Identifier
eUICC	Embedded Universal Integrated Circuit Card
FCM	Firebase Cloud Messaging
FQDN	Fully Qualified Domain Name
GCM	Google Cloud Messaging
GID1	Group Identifier 1 as defined in TS 31.102

Abbreviation	Definition
GID2	Group Identifier 2 as defined in TS 31.102
HTTP	Hyper-Text Transfer Protocol
HTTPS	Hyper-Text Transfer Protocol Secure
ICCID	Integrated Circuit Card Identifier
IMEI	International Mobile Equipment Identity
IMS	IP Multimedia Subsystem
IMSI	International Mobile Subscriber Identity
JSON	JavaScript Object Notation
JWT	JSON Web Token
LPA	Local Profile Assistant
LTE	Long-Term Evolution
MCC	Mobile Country Code (As defined in E.212)
MDM	Mobile Device Management
MNC	Mobile Network Code (As defined in E.212)
MO	Management Object
MSISDN	Mobile Subscriber Integrated Services Digital Network Number
ODSA	On-Device Service Activation
OIDC	OpenID Connect
OMNA	Open Mobile Naming Authority, registry available at: http://www.openmobilealliance.org
OTP	One-Time Password
PRD	Permanent Reference Document
RCS	Rich Communication Services
SIM	Subscriber Identity Module
SMS	Short Message Service
SMSoIP	SMS Over IP
SNC	Service Notification Channel
SP	Service Provider
TAD	Technical Adaptation of Devices
TLS	Transport Layer Security
T&C	Terms & Conditions
UDH	User Data Header
URL	Uniform Resource Locator
VoWiFi	Voice-over-WiFi
VoLTE	Voice-over-LTE
VoNR	Voice-over-New-Radio
WNS	Windows Push Notification Service

Abbreviation	Definition
XML	Extensible Markup Language
XSD	Extensible Markup Language Schema Definition

1.5 Definitions

Definition	Meaning
Client	Component/module on a device that provides the Voice-over-Cellular or VoWiFi service. A client verifies with the network's Entitlement Configuration Server if it is entitled or not to offer that service to end-users.
Entitlement	The applicability, availability, and status of a service, needed by the client before offering that service to end-users.
Entitlement Configuration	Information returned to the client by the network, providing entitlement information on a service.
Entitlement Configuration Server	The network element that provides entitlement configuration for different services to clients.

1.6 References

Ref	Document Number	Title
[1]	OMA-APPIDREG	OMA Registry of Application Identifiers (AppID) http://www.openmobilealliance.org/wp/OMNA/dm/dm_ac_registry.html
[2]	IR.51	GSMA PRD IR.51 - "IMS Profile for Voice, Video and SMS over untrusted Wi-Fi access" Version 5.0, 23 May 2017. http://www.gsma.com
[3]	IR.92	GSMA PRD IR.92 - "IMS Profile for Voice and SMS" Version 15.0, 14 May 2020. http://www.gsma.com
[4]	NG.102	GSMA PRD NG.102 - "IMS Profile for Converged IP Communications" Version 6.0, 13 April 2019. http://www.gsma.com
[5]	RCC.14	GSMA PRD RCC.14 "Service Provider Device Configuration", Version 10.0, 04 June 2024. http://www.gsma.com
[6]	RFC2119	"Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997. http://www.ietf.org/rfc/rfc2119.txt
[7]	TS.22	Recommendations for Minimum Wi-Fi Capabilities of Terminals, Version 6.0, 14 December 2018. http://www.gsma.com
[8]	TS.32	Technical Adaptation of Devices through Late Customisation, Version 7.0, 20 April 2020. http://www.gsma.com
[9]	E.212	Mobile network codes (MNC) for the international Identification plan for public networks and subscriptions (according to recommendation ITU-T E.212 (05/2008))
[10]	SGP.21	Remote SIM Provisioning Architecture. http://www.gsma.com
[11]	SGP.22	Remote SIM Provisioning Technical Specification. http://www.gsma.com

[12]	RFC2616	Hypertext Transfer Protocol HTTP/1.1 IETF RFC, http://tools.ietf.org/html/rfc2616
[13]	RCC.07	GSMA PRD RCC.07 “Rich Communication Suite - Advanced Communications Services and Client Specification”, Version 11.0, 16 October 2019. http://www.gsma.com
[14]	OpenID Connect	OpenID Connect Core; OpenID Foundation http://openid.net/connect/
[15]	RFC6749	The OAuth 2.0 Authorization Framework. https://tools.ietf.org/html/rfc6749
[16]	RFC7521	Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants. https://tools.ietf.org/html/rfc7521
[17]	RFC7523	JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants. https://tools.ietf.org/html/rfc7523
[18]	RFC4187	Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA). https://tools.ietf.org/html/rfc4187
[19]	3GPP TS 23.503	Policy and Charging Control Framework for the 5G System. http://www.3gpp.org
[20]	3GPP TS 24.526	User Equipment (UE) policies for 5G System (5GS) http://www.3gpp.org
[21]	3GPP TS 31.102	Characteristics of the USIM Application http://www.3gpp.org
[22]	RFC3986	Uniform Resource Identifier (URI): Generic Syntax. https://tools.ietf.org/html/rfc3986
[23]	ISO/IEC 18004:2015	Information technology -- Automatic identification and data capture techniques -- QR Code bar code symbology specification
[24]	IEEE 1003.1-2017	IEEE Standard for Information Technology--Portable Operating System Interface (POSIX(R)) Base Specifications, Issue 7

1.7 Conventions

“The key words “must”, “must not”, “required”, “shall”, “shall not”, “should”, “should not”, “recommended”, “may”, and “optional” in this document are to be interpreted as described in [6].”

Any parameter defined as **String** type along this document must be considered as **case insensitive** for any comparation operation.

2 Entitlement Configuration Procedures

2.1 Default Entitlement Configuration Server

The client may follow a discovery procedure to obtain the address of the entitlement configuration server. The resulting FQDN may be based on the following format:

- aes.mnc<MNC>.mcc<MCC>.pub.3gppnetwork.org

Whereby <MNC> (Mobile Network Code) and <MCC> (Mobile Country Code) shall be replaced by the respective values of the home network in decimal format and with a 2-digit MNC padded out to 3 digits by inserting a 0 at the beginning.

The details of the discovery procedure and resulting address are part of the agreements between carriers and Entitlement Client providers and are out of scope of this specification.

2.1.1 eSIM metadata containing the Entitlement Configuration Server parameters

The eSIM profile metadata may contain a `VendorSpecificExtension` in `serviceSpecificDataStoredInEuicc` defined in SGP.22 [11]. The structure of this object is defined as follows:

```
vendorOid gsmaTs43Oid OBJECT IDENTIFIER

ServiceProviderTs43Config ::= SEQUENCE{  -- Tag 'xxxx'
serviceProviderTs43Capabilities [x] ServiceProviderTs43Capabilities, - Tag 'xxxx'
}
ServiceProviderTs43Capabilities ::= SEQUENCE of SEQUENCE{ -- Tag 'xxxx'
      entitlementServerFqdn [x] UTF8String (SIZE(0..64)), -- Tag 'xxxx'
}
```

The ServiceTs43ProviderCapabilities object will contain the following:

SGP.22 Object Name	Type	Description
entitlementServerFqdn	UTF8String	The FQDN of the ECS the client application can send requests to.

Table 3. Objects contained in the ServiceTs43ProviderCapabilities

The client application may use this information in order to configure its ECS parameters associated with the eSIM profile.

2.2 HTTP Headers

2.2.1 User-Agent HTTP header

The client shall include the User-Agent header in all HTTP requests. The `User-Agent` header should be compiled as defined in RCC.07 [13] section C.4.1 “User-Agent and Server Header Extensions” including the following amendment:

```
product-list =/ enabler *(LWS enabler)
                    [LWS terminal]
                    [LWS client]
```

```
                    [LWS OS]
```

The rule “enabler” is defined in RCC.07 [13] and extended as:

```
enabler =/ GSMA-PRD-TS43 ; GSMA PRD reference
GSMA-PRD-TS43 = "PRD-TS43"
```

The rule “client” is defined in RCC.07 [13] and extended as:

```
client =/ "client-" client-ts43 SLASH client-ts43-version
client-ts43 = "IMS-Entitlement" / "Companion-ODSA" / "Primary-ODSA" / "Server-ODSA"
client-ts43-version = alphanum *15(alphanum / "." / "-");version identifying the
client,
```

The rules “terminal” and “OS” are those defined in RCC.07 [13] section C.4.1

- Examples:

```
User-Agent: PRD-TS43 term-Vendor1/Model1-XXXX client-IMS-Entitlement/1.0 OS-
Android/8.0

User-Agent: PRD-TS43 term-Vendor1/Model1-XXXX client-Companion-ODSA/1.55B.devkey-20
OS-Android/10.0

User-Agent: PRD-TS43 term-Vendor1/Model1-XXXX client-Primary-ODSA/dev20200812 OS-
Other/0.4
```

Where `XXXX` is a 20 characters max string identifying the model.

2.2.2 Accept-Language HTTP header

The client application and ECS shall support Accept-Language for local language support as defined in RCC.14 [5]. This is to make certain that any user readable messages sent to the client can be localized to the language set in the header.

2.3 HTTP GET method Parameters.

A client supporting service entitlement configuration shall indicate the support by inclusion of an "app" HTTP GET request parameter as defined in RCC.14 [5] with the proper identifiers for the targeted entitlement.

The Open Mobile Naming Authority (OMNA) maintains a registry of values for Application Characteristic Identifier (AppID) and the range ap2001-ap5999 is used for externally defined Application entities. The following AppIDs[1] are used for VoWiFi, Voice-over-Cellular, SMSoIP and Direct Carrier Billing entitlement applications, and for the ODSA for Companions, Primaries and Server to Server applications:

- Voice-over-Cellular Entitlement - AppID of “ap2003”
- VoWiFi Entitlement - AppID of “ap2004”
- SMSoIP Entitlement – AppID of “ap2005”
- ODSA for Companion device, Entitlement and Activation – AppID of “ap2006”
- ODSA for Primary device, Entitlement and Activation – AppID of “ap2009"

[1] AppIDs are obtained from OMA by contacting mailto:helpdesk@omaorg.org and supplying the information requested here https://www.openmobilealliance.org/wp/OMNA/dm/dm_ac_registry.html

- Data Plan Related Information Entitlement Configuration - AppID of “ap2010"
- ODSA for Server Initiated Requests, Entitlement and Activation – AppID of “ap2011”
- Direct Carrier Billing – AppID of “ap2012”
- Private User Identity – AppID of “ap2013”
- Device and User Information – AppID of “ap2014”
- App authentication – AppID of “ap2015”
- SatMode Entitlement – AppID of “ap2016”

The parameters from RCC.14 [5] (“IMSI”, “`token`”, “`vers`”, “`app`”, “`GID1`”, “`GID2`”, “`terminal_vendor`”, “`terminal_model`”, “`terminal_sw_version`”) are used for entitlement configuration requests but some have been specifically redefined in Table 4 in order to remove the length limits imposed in that spec. In addition, new parameters are introduced specific for entitlement purposes, as described in Table 4.

HTTP GET parameter	Type	Description	Usage
terminal_id	String	This value shall be a unique and persistent identifier of the device. This identifier may be an IMEI (preferred) or a UUID.	Required.
requestor_id	String	This value shall be a unique and persistent identifier of the system interacting with ECS. If the requestor_id is present in the request, the terminal_id will become optional.	Required in those scenarios where the system triggering the request acts on behalf of the primary device. Examples of these systems are MDM or Application Server.
entitlement_version	String	GSMA PRD version implemented by the client. Set to this current version, or earlier one (see section 2.5). entitlement_version parameter will map with any existing document history version (without ‘V’ if there were any). This version number is expected to be defined as the following ABNF rule: 1*DIGIT”.”1*DIGIT. Some valid entitlement versions are: 6.0 ; 6.1 ; 10.0 or 11.10	Required.
app_name	String	The name of the device application making the request.	Optional. (see section 2.8.5 for recommended values)
app_version	String	The version of the device application making the request.	Optional.

HTTP GET parameter	Type	Description	Usage
notif_token	String	The registration token to be used when notifications are transmitted to the device over a cloud-based messaging infrastructure (refer to 2.6).	Optional, required each time the device obtains or disables a registration token from the notification service. Sent at the same time as "notif_action" parameter.
notif_action	Integer	The action associated with the registration token "notif_token" parameter. Possible values are: • 0 - disable notification token • 1 - enable GCM notification token • 2 - enable FCM notification token • 3 - enable WNS push notification • 4 - enable APNS notification token • 5 - enable SNC notification token	Optional, required if the "notif_token" parameter is present.
temporary_token	String	A token to be use instead of the TOKEN if available to the client application.	Conditional to the client application not having TOKEN to authenticate with the ECS.
operator_token	String	A token to be use instead of the TOKEN if available to the client application.	Conditional to the client application not having TOKEN to authenticate with the ECS.
terminal_vendor	String	This field identifies the terminal OEM.	Required.
terminal_model	String	This field identifies the terminal model.	Required.
terminal_sw_version	String	This field identifies the terminal software version.	Required.

Table 4. GET Parameters for Entitlement Configuration Request

Entitlement use cases can also define its own set of request parameters. Refer to 6.2 for the parameters associated with the Companion and Primary ODSA use cases.

Table 5 presents a sample HTTP GET request for VoWiFi entitlement with the parameters located in the HTTP query string.

```
GET ? terminal_id = 013787006099944&
token = es7w1erXjh%2FEC%2FP8BV44SBmVipg&
terminal_vendor = TVENDOR&
terminal_model = TMODEL&
terminal_sw_version = TSWVERS&
entitlement_version = ENTVERS&
app = ap2004&
GID1 = 123D&
vers = 1 HTTP/1.1

Host: entitlement.telco.net:9014
User-Agent: PRD-TS43 TVENDOR/TMODEL IMS-Entitlement/TSWVERS OS-Android/8.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Table 5. Example of an HTTP GET Entitlement Configuration Request

2.4 HTTP POST Method

In addition to the HTTP GET, the HTTP POST method can be used by the client for entitlement configuration request. In this case, the parameters are located in the HTTP message body and should follow the JSON object value format. The same parameters defined in section 2.3 are used for the POST request.

If a client supports the POST method, it shall use it instead of the GET method for entitlement configuration requests. The Entitlement Configuration Server should be able to process both GET and POST methods. In case the server does not support POST, it shall return an HTTP response with 405 “Method Not Allowed”. In that case, the client should resend the request using the GET method.

The message body of the HTTP POST request follows the content type of "`application/json`" and is provided as a JSON object value (it is not encoded). The resulting HTTP response can be encoded as described in 2.9.1.

Table 6 presents a sample HTTP POST request for VoWiFi entitlement with the parameters located in the HTTP message body.

```
POST / HTTP/1.1
Host: entitlement.telco.net:9014
User-Agent: PRD-TS43 TVENDOR/TMODEL IMS-Entitlement/TSWVERS OS-Android/8.0Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Content-Type: application/json

{
      "terminal_id" : "013787006099944",
      "entitlement_version" : "ENTVERS",
      "token" : "es7w1erXjh%2FEC%2FP8BV44SBmVipg",
       "terminal_vendor" : "TVENDOR",
       "terminal_model" : "TMODEL",
       "terminal_sw_version" : "TSWVERS",
       "app" : "ap2004",
       "vers" : "1"
}
```

Table 6. Example of an HTTP POST Entitlement Configuration Request

As described in reference [5] app parameter could be multi-valued. Unlike how this multi-valued parameter is sent when using GET method (and string concatenating app=appID with '&' character), in case of POST method, `AppID` values will be sent as an array of strings.

Example: `"app" : ["ap2003", "ap2004", "ap2005"]`.

In case of a single AppID value, a single string value (instead of an array with a single string) will be expected.

Example: `"app" : "ap2003"`

2.5 Protocol version control

As clients and servers may support different versions of the same protocol, a control phase is required. The main rules for this check are:

- The client indicates the supported protocol version in the parameter "`entitlement_version`".
- The server shall answer accordingly to the request if it supports the version indicated in the parameter, or it shall return a 406 "Not Acceptable" response when it does not, including a Reason-Phrase set to "protocol not supported".

2.6 Network Requested Entitlement Configuration

Two mechanisms are available to operators to trigger an entitlement configuration request from a device application, either:

- by sending a Short Message Service (SMS) message to the target device, or
- by sending a notification message to the device over a cloud-based messaging infrastructure (APNS, FCM, GCM, WNS or SNC)

When an application is notified in this manner, it shall generate the proper Service Entitlement request to the entitlement configuration server:

- For applications “ap2003”, “ap2004”, “ap2005”, “ap2010” and “ap2016” (Voice-over-Cellular, VoWiFi or SMSoIP entitlement, DataPlan, SatMode entitlement) a GET or POST HTTP request for the corresponding app is generated.
- For applications “ap2006” or "ap2009" (ODSA for Companion or Primary device), a GET or POST HTTP request for the corresponding app and the operation of AcquireConfiguration is generated.

2.6.1 SMS-Based Notifications

To notify the target device of a change in the entitlement configuration, the entitlement configuration server can use the same method described in Chapter 3 of RCC.14 [5] and generate a Short Message Service (SMS) message towards the target device via application-port addressed SMS with a User Data Header (UDH).

The User Data Header (UDH) contains the following Information Elements:

- **UDH length**: 6 (six octets)
- **Information-Element-Identifier** (IEI): x05, message is using "application port addressing scheme, 16-bit address”.
- **Destination application port**: by default, set to 8095 or 0x1F9F
- **Source application port**: set to 0

The content of the message is different from RCC.14 [5], in order to differentiate a network-triggered notification coming from a configuration server and one coming from an entitlement configuration server:

- Instead of the SMS user-data set to: `user-id “-rcscfg” [ “,” param ]`
- The following is used: `user-id “-aescfg” [ “,” param ]`

The “parm” parameter contains the application(s) notified with this SMS. An example of the SMS content is:

```
214011001388741-aescfg,ap2003
```

This message would trigger (or wake up) the Voice-over-Cellular application on the device to create and send a request (HTTP GET with service parameters) to the Entitlement Configuration Server. If several applications are targeted, they would appear as a comma-separated list, for example:

```
214011001388741-aescfg,ap2003,ap2004,ap2005
```

2.6.2 Messaging Infrastructure-Based Notifications

A notification message can also be sent by the Entitlement Configuration Server to the device over a cloud-based messaging infrastructure that devices registered with in order to receive network-initiated messages. The device’s application is reached and identified via the notif_token present in the original GET request received by the entitlement configuration server.

The details of the cloud-based messaging technology, including the contained values in the payload, are implementation dependent and not covered in this specification. The payload of

the notification message is a JSON object value that should contain a "**`data`**" element with at least two key-value pairs:

- "**app**": the application targeted for re-configuration, all **app**s are eligible for the service, examples of **app**s:""`ap2003`", "`ap2004`","`ap2005`", "`ap2006`", "`ap2009`" and others. If multiple applications are targeted, the value is a JSON array of strings.
- "**timestamp**": the time of the notification, in ISO 8601 format, of the form `YYYY-MM-DDThh:mm:ssTZD`, where `TZD` is time zone designator (Z or `+hh:mm` or `-hh:mm`).
- An example of the notification payload for Voice-over-Cellular follows:

```
"data":
  {
     "app": "ap2003",
     "timestamp": "2019-01-29T13:15:31-08:00"
  }
```

- An example of the notification payload for multiple applications follows:

```
"data":
{
   "app": ["ap2003", "ap2004", "ap2005"],
   "timestamp": "2019-01-29T13:15:31-08:00"
}
```

2.7 Roaming Conditions

The fact that the device is roaming does not impact the ability of a client to request an entitlement configuration. The client can send the HTTP-based entitlement configuration request over an available data connection, either Wi-Fi or a cellular data APN. Refer to NG.102 [4] for the configuration and usage of those connections as related to operator traffic.

The device can therefore be in a roaming situation when requesting for an entitlement configuration on Voice-over-Cellular and/or VoWiFi.

2.8 Authentication Mechanism

The different authentication procedures described in of RCC.14 [5] shall be followed during the entitlement configuration exchange.

Entitlement configuration is usually triggered by the device or client and the user is not aware of an entitlement configuration process taking place. It is then preferable for the entitlement configuration server to rely on authentication mechanisms like "User Authentication via HTTP Embedded EAP-AKA" which does not involve user interactions.

In case access to the device's SIM data is not possible (which would prevent authentication based on EAP-AKA) or the client encounters a failure at the ECS, authentication following the OpenID or OAuth 2.0 procedure is the preferred alternative.

Both authentication methods are detailed in the following two sections.

2.8.1 Embedded EAP-AKA Authentication by Entitlement Configuration Server

The Embedded EAP-AKA procedure of RCC.14 [5] involves a separate authentication server included in the flow as part of an HTTP Redirect response (as per OpenID Connect). In case an operator does not carry such OpenID Connect authentication server with EAP relay capabilities and its entitlement configuration server supports the EAP relay function, it is possible for the server to omit the HTTP Redirect and exchange the EAP payloads directly with the client.

This flow is shown in Figure 2. Note that the EAP payload specification along with the GET and POST headers and parameters defined in RCC.14 [5] for the HTTP Embedded EAP-AKA procedure of RCC.14 [5] are kept. The only difference is the omission of the HTTP 302 Found responses (HTTP redirects).

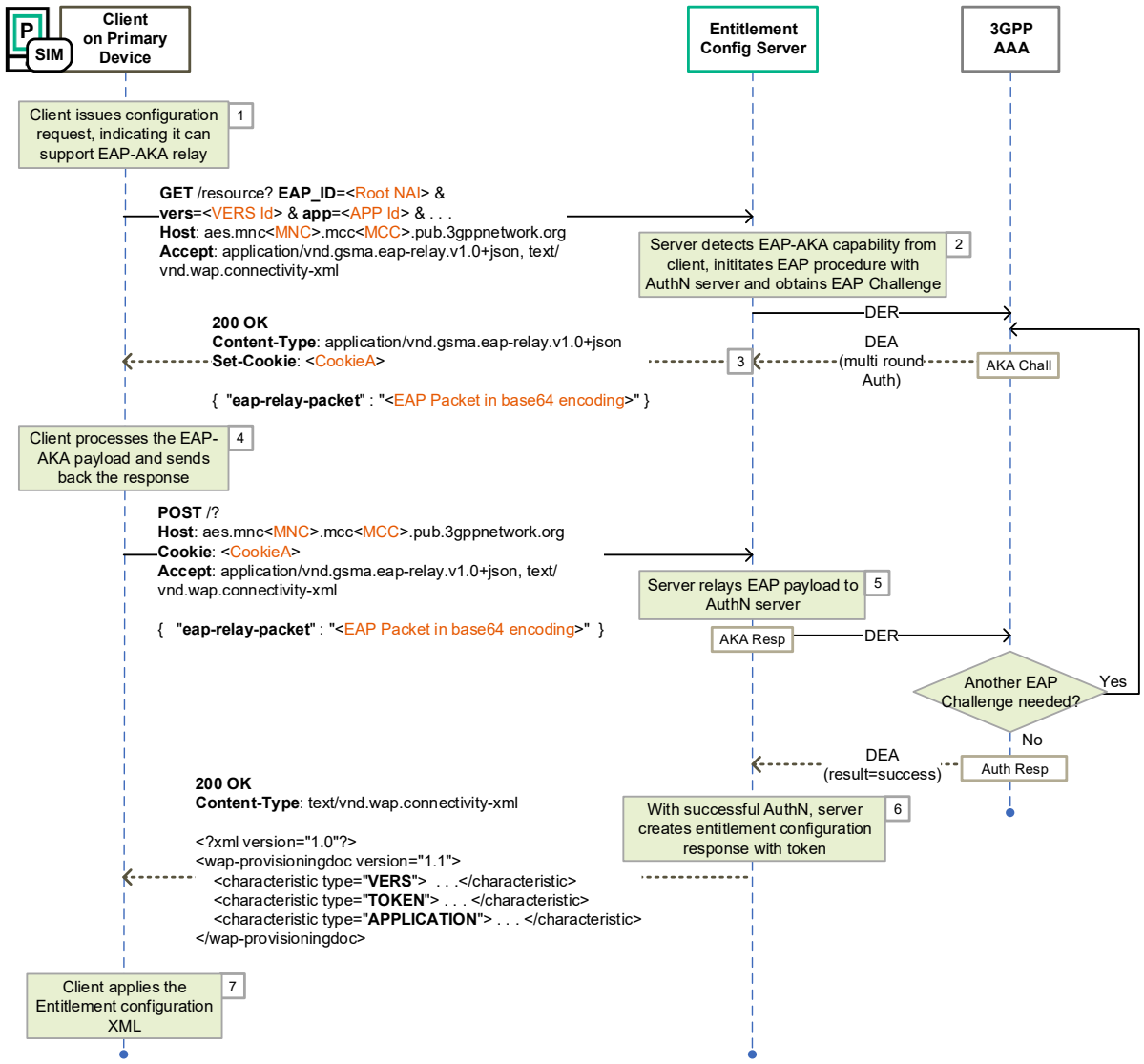

Figure 2. Embedded EAP-AKA Authentication Flow with Entitlement Configuration Server Supporting EAP Relay Function

If the Entitlement Configuration Server is handling the EAP-AKA relay to an operator's Authentication server (a 3GPP AAA for example), Table 7 shows the mapping between the response codes from the 3GPP AAA and the corresponding HTTP GET response. The response code is coming from AVP « Result-Code » or AVP « Experimental-Result » sent by the 3GPP AAA in the Diameter EAP Response (DER).

DER Result Code	HTTP GET Response	Reason
1001	200 OK	Waiting for AKA challenge response from device
2001	200 OK	Successfully authenticated by AAA
3001-3010, 5002, 5004-5017 Connection failure to 3GPP AAA	**If ECS and the application supports alternate forms of authentication, and the client did not include a TOKEN in the original request:** 511 Network Authentication Required **Otherwise:** 503 Retry After / Service Unavailable	Connectivity, protocol errors and miscellaneous AAA errors, which could be transient, can be resolved by retrying (503) or by indicating to the client that an alternate form of authentication is available (511).
4001, 5001, 5003	**403 Forbidden**	As the Identity is unknown to the AAA (4001 DIAMETER_AUTHENTICATION_REJECTED, 5001 DIAMETER_ERROR_USER_UNKNOWN, and 5003 DIAMETER_ERROR_IDENTITY_NOT_REGISTERED) the failure is permanent and requires some action on either the device (to change identities) or on AAA to populate said identities.

Table 7. Mapping Between 3GPP AAA's DER Result Code and HTTP Response Code

The way the Entitlement Configuration Server manages the calls to the AAA is out of scope of this document. It is possible for the client on the device to perform a request with EAP_ID parameter as shown in Figure 2 along with a valid token parameter. In this case, the Entitlement Configuration Server may either:

- perform a full EAP AKA authorization based upon the EAP_ID parameter only.
- check the token validity and avoid requesting the AAA.

2.8.2 Authentication with OAuth 2.0 / OpenID Connect Procedure

The OpenID Connect (OIDC) authentication method is available for clients that cannot access the AKA function of the SIM on the device and the Service Provider decides not to use other Authentication methods like SMS-OTP. The end-user must instead go through an authentication procedure managed by the Service Provider's OAuth 2.0 / OIDC authentication server. The invocation of OIDC-based authentication by the entitlement configuration server follows the procedure defined in section 2.8 of RCC.14 [5].

Figure 3 presents an overview of the steps for the OIDC-based authentication procedure, shown here for informational purposes.

- After deciding that OIDC procedure is needed (lack of `token` or invalid `token`, no `EAP_ID` in GET request, other authentication methods such as EAP-AKA not supported), the entitlement configuration server redirects (with 302 Found) the GET request from the device's client to the Service Provider's OIDC authentication endpoint.
- The OIDC authentication endpoint can offer different types of authenticators, some of which involve actual user actions.
- When the end-user is authenticated, the entitlement configuration server will receive an OAuth 2.0 "auth code" from the authentication server (via the client or user agent on the device, again using a 302 Found).
- The entitlement configuration server requests for both an access token and an ID Token from the Service Provider's OIDC Token endpoint.
- After validating the OAuth 2.0 access token and the OpenID token, the entitlement configuration server can identify the end-user subscription and resumes processing of the original GET resource request.

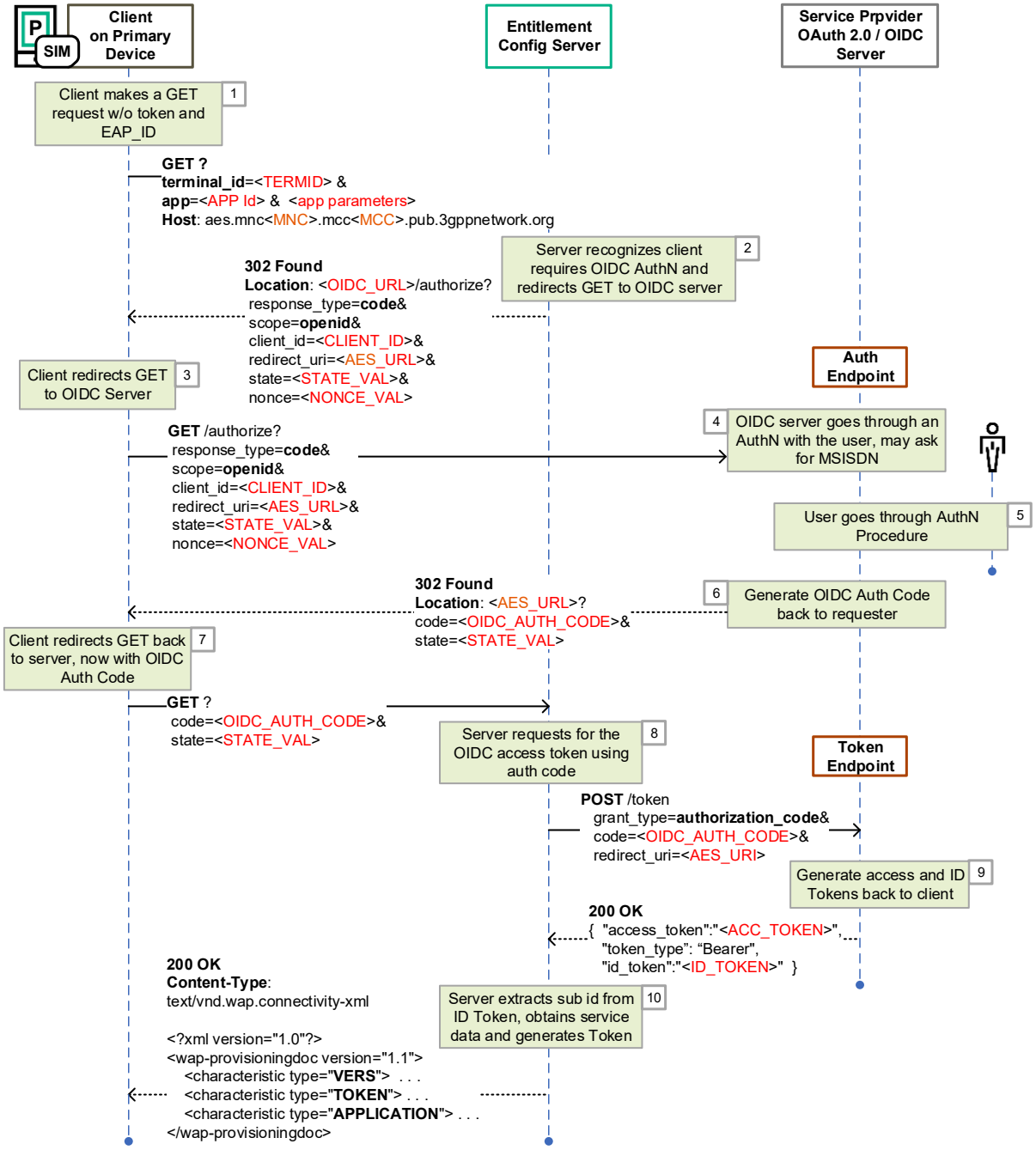

Figure 3. OAuth 2.0 / OpenID Authentication Flow with Entitlement Configuration Server

2.8.2.1 Authentication with OAuth 2.0/OpenID Connect and SMS-OTP as second authentication factor.

This extension of the authentication described in the previous section 2.8.2 describes a way to use SMS-OTP to provide a second authentication factor. This may be useful when a single factor OIDC authentication does not provide enough guarantees (e.g.: a login/password does not assure that the associated mobile device is the one performing the TS43 requests). In this example, it is assumed the client expects an xml format. The SMS-OTP method described in this section is an example and may be replaced with any other method as long as it brings enough guarantees.

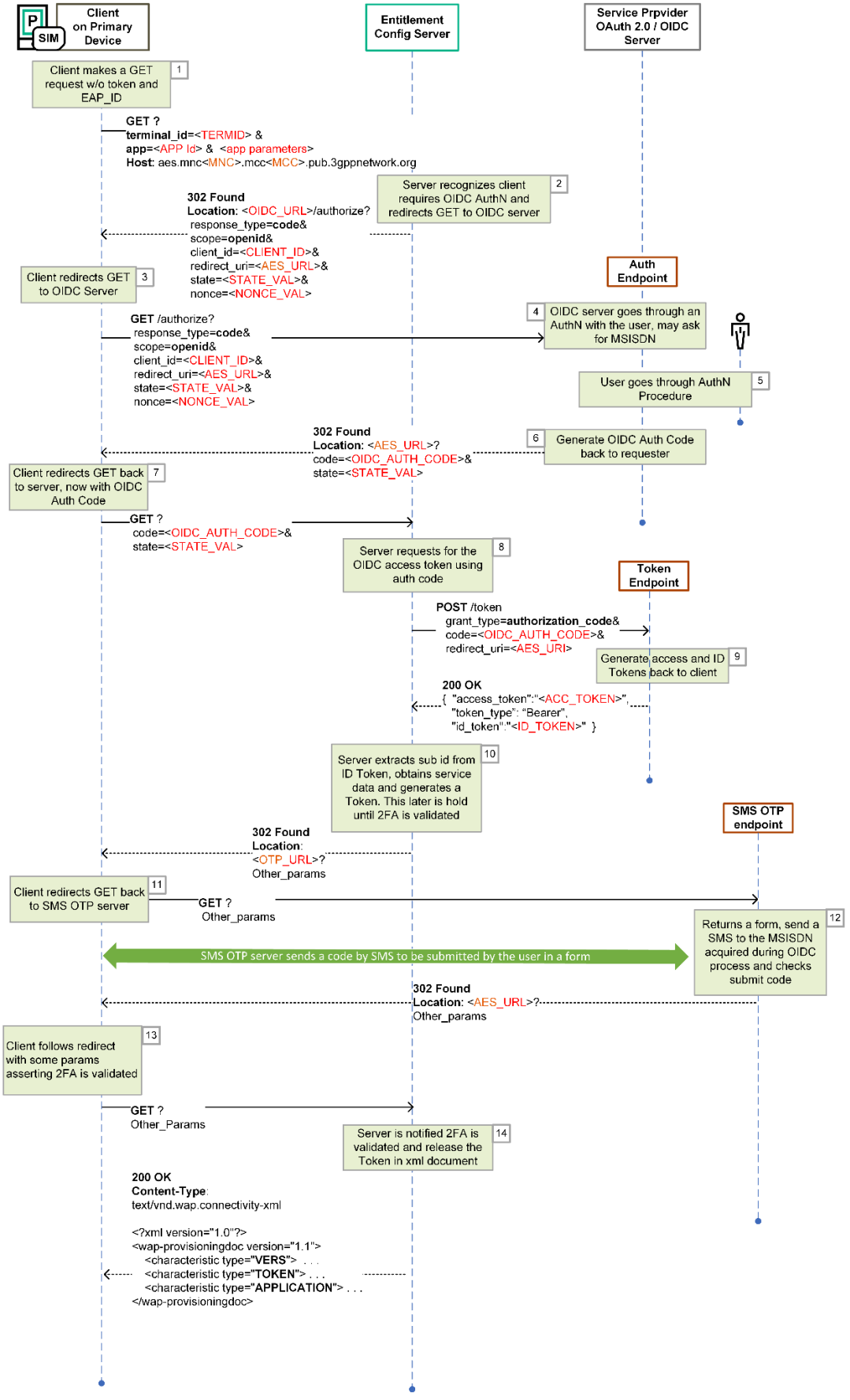

Figure 4. – OAuth 2.0 / OpenID Authentication Flow and SMS-OTP with Entitlement Configuration Server

Steps 1 to 9 are identical to section 2.8.2. Next steps are:

10. The ECS returns another HTTP 302 redirection towards the SMS-OTP server.
11. The client redirects GET back to the SMS-OTP server which generates a code and returns a form. The exchanges between the client and OTP endpoint are not described in this example. The objective is to exchange a code to check the user has the mobile from which the OIDC authentication is performed.
12. Once second authentication factor is checked, the SMS-OTP server returns a 302 redirection towards the ECS to resume the sequence in section 2.8.2.
13. The client follows the redirection with some params allowing the ECS to return the Token. The final step 14 is identical to step 10 of the previous section 2.8.2.

In this example, it is important to note that in the client perspective, the steps 7 & 13 look very similar as the input is a 302 redirect towards the ECS (though with different parameters), but the outcome is very different (webview vs xml document).

2.8.3 Server to Server Authentication using OAuth 2.0 server and JWT.

The server-to-server authentication using OAuth2.0 is available for server applications (client) that needs to access a service without any user interaction.

The authentication flow described in this section follows the architecture described in Figure 5 where (as defined in reference [15] – *1.1 Roles*) the different roles are (between brackets how it is mapped to the systems involved in this TS.43 spec):

- ***Resource Owner*** [*server managing devices – aka MDM –*]. An entity capable of granting access to a protected resource. In the scope of this
- ***Client*** [*server ODSA App*]. An application making protected resource requests on behalf of the resource owner and with its authorization.
- ***Resource Server*** [*ODSA Device GW – Entitlement Configuration Server*]. The server hosting the protected resources, capable of accepting and responding to protected resource requests using access tokens.
- **Authorization server** [*Service Provider's OAuth2.0 server*]. The server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorization.

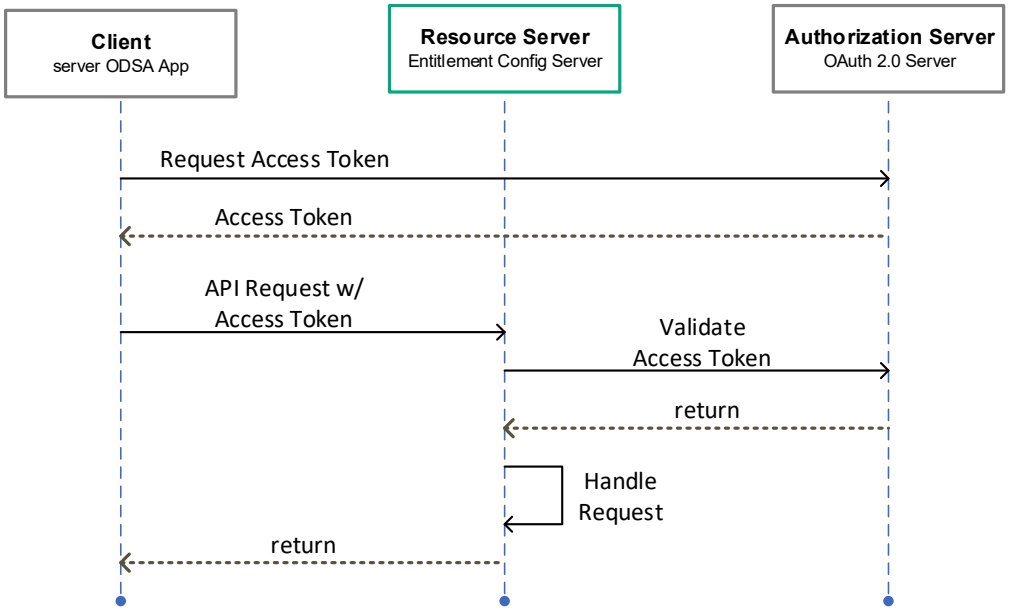

Figure 5. Client Authentication Flow (server to server AuthN using OAuth2.0)

Previously to send and Access Token Request, it is necessary that the Client gets `client_id` and `client_secret` from the Authorization Server. The process to obtain these two parameters are not covered in this specification.

Client applications have an attribute named `client_type` (see reference [15] – *2.1 Client Types*) and when this `client_type` is confidential (as it is in our case) the client authentication is required to get the access token.

Among the different method to perform the Client Authentication, the one using JSON Web Token (JWT) is the selected one for this specification (see reference [17] for additional info). This method does not require the `client_secret` to be sent in the request at all but it is used to sign the JWT.

In the context of client authentication, the JWT is called client assertion. The access token request requires (as defined in reference [16] - *4.2 Using Assertions for Client Authentication*) `client_assertion_type` (the value is the following fixed string, `urn:ietf:params:oauth:client-assertion-type:jwt-bearer`) and `client_assertion` (the JWT containing the information for client authentication).

The JWT (as defined in reference [17] - *3 JWT Format and Processing Requirements*) payload must contain (at least):

- **iss** (issuer). It contains a unique identifier for the entity that issued the JWT. It should be the `client_id`.
- **sub** (subject). It identifies the principal that is the subject of the JWT. It should be the `client_id`.
- **aud** (audience). It contains a value that identifies the authorization server as an intended audience. It should be the URL of the authorization server.
- **exp** (expiration time). It indicates the time window during which the JWT can be used.

Figure 6 presents an overview of the steps for the Client authentication (server to server) procedure to get the access token. The validation of this access token is described in each process where this authentication takes place.

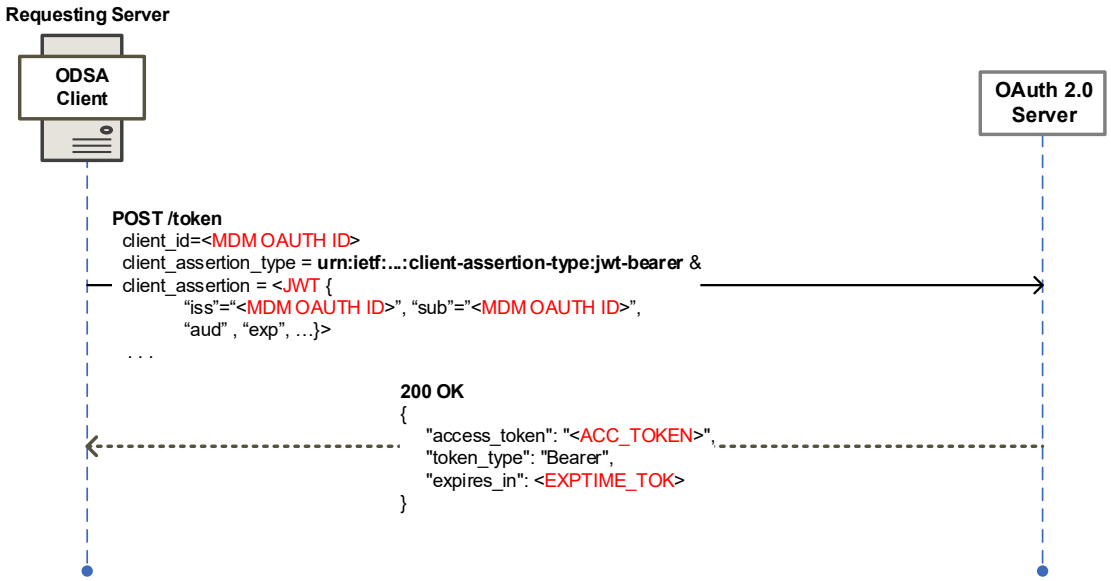

Figure 6. Getting Access Token in Client Authentication Flow

2.8.4 Error processing

Some errors might occur during the OIDC user authentication procedure, see OpenID Connect [14] section Authentication Error Response. For example, the user could decline a consent screen, or the Open Id Connect server could get a technical issue (e.g. invalid request).

For the user to be presented an ad-hoc explanation page related to an authentication error, the ODSA entitlement parameters **GeneralErrorURL** and **GeneralErrorUserData** are defined in section 6.5.1 allowing the client application to interact with the Service Provider's portal web server.

The Figure 7 presents an overview of the steps for the OIDC-based authentication procedure in case of error, shown here for informational purposes.

Steps 1-4 are similar to those described in previous section. In the next steps:

5. The user does not succeed to complete the OIDC-based authentication procedure.
6. The Service Provider's OIDC authentication endpoint returns to the Client the redirection URI specified in the Authorization Request with the appropriate error and state parameters.
7. The client on primary device redirects the error URL to the Entitlement Server.
8. The Entitlement Server generates an XML document as a 200 OK answer. This document does not embed a token, as the opposite of the successful case, but an

URL and data to be used by the client (parameters `GeneralErrorURL` and `GeneralErrorUserData`).

9. The client is notified of the error thanks to the presence of these parameters in the document and displays the error webview referenced by the `GeneralErrorURL`, using the `GeneralErrorUserData` in the query string.

10. The end user closes the webview, activating the `dismissFlow` callback.

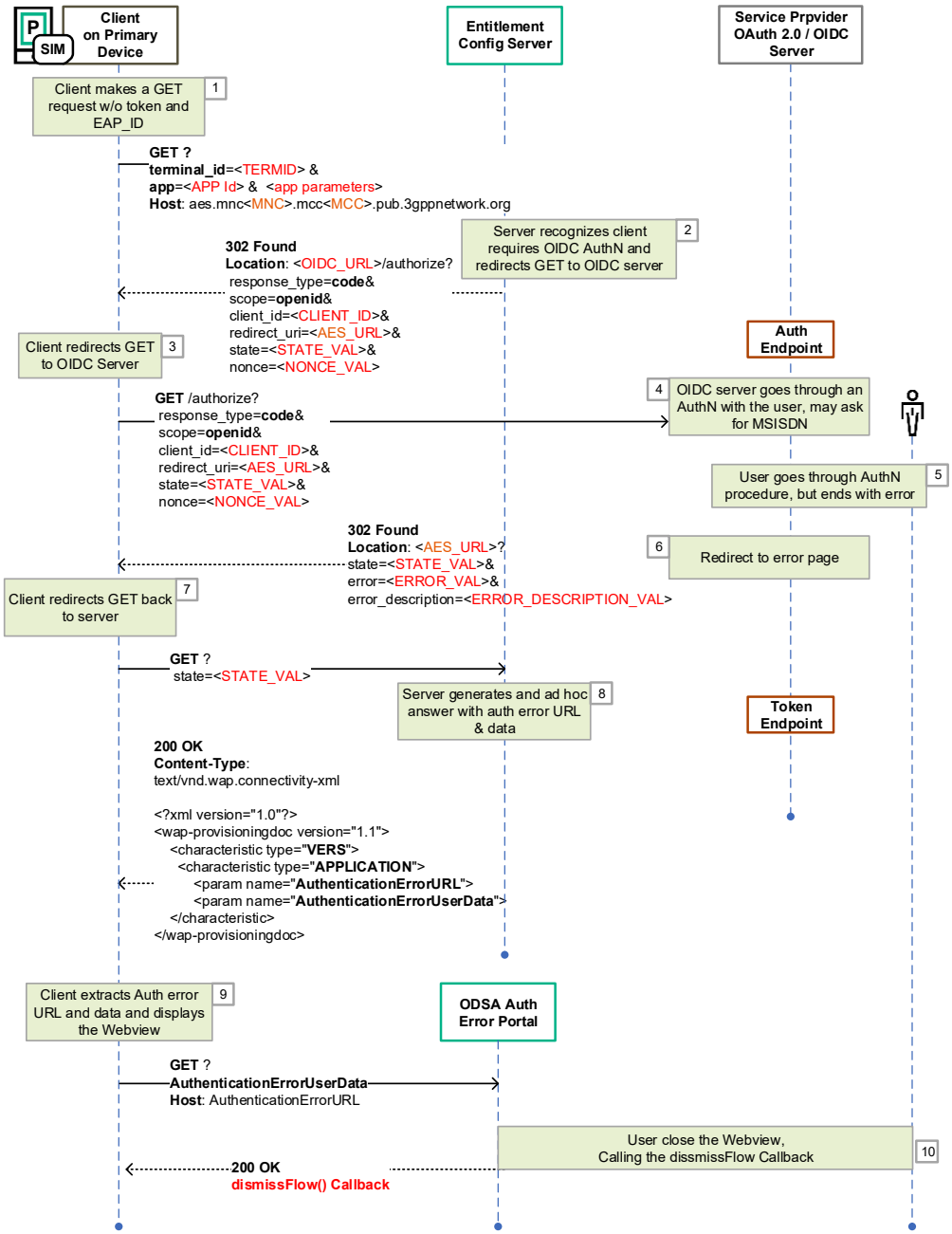

Figure 7. OAuth 2.0 / OpenID Authentication Error Flow with Entitlement Configuration Server

2.8.5 Fast Authentication and Token Management

Authentication is one of the key pillars in the service entitlement configuration protocol and requires that any system/device interacting with ECS performs this authentication. There are different ways to authenticate the device as defined in the sections 2.8.1 (EAP-AKA), 2.8.2 (OAuth 2.0 / OIDC) and 2.8.3 (server to server authentication using OAuth 2.0). All these authentication mechanisms require to interact with additional elements in the network as it could be AAA and/or OAuth 2.0 / OIDC server. To avoid an extra load in each authentication request, ECS shall implement a fast authentication method using an internal token that is managed between device and ECS. If device includes in the request a valid AuthN Token generated by ECS in the previous request, it should not be necessary to perform a full authentication (through AAA and/or OAuth 2.0 / OIDC). Device will be authenticated by AuthN Token.

Authentication Token generated by ECS shall have an expiration time (at this point device will need to perform a full authentication) and token could be regenerated in each device request/interaction.

ECS should implement the logic to avoid, as far as possible, that different clients in the same device, using the same AppID, could overwrite each other the AuthN token generated for the other client. As a recommendation, ECS could use the following parameters to identify the client in a device using a specific app: `terminal_id`, `app` and `app_name`.

Note- Due to `app_name` is an optional attribute in the request, and is not standardized, it is recommended that the AppID client tries to define an `app_name` as unique identifier, including some code (characters) that could be considered specifically for that AppID developer (vendor identifier, carrier identifier, …).

2.8.6 Token Management for Temporary Tokens

Temporary Tokens are used when the established trust between ECS and the client should be extended to a third party by creating a temporary token for a specific action. When a temporary token is used, the token handling procedures differ from the regular used `auth_token`.

A temporary token is handed out by the ECS to the client, passed on to a third party (e.g. Application Server) and used by that third party for authentication on the ECS. This implies that error messages will be exchanged between the third party and the ECS.

The communication between the client and the third party is outside of the scope of TS.43.

The communication between the third party and the ECS using temporary Token should follow the Error Handling cases described in Table 8:

For implementations running automated retries, a backoff mechanism should be used.

Scenario	GET/ POST Response Code from ECS	3rd Party Server Action
Invalid or missing parameters or wrong format in Request	**400** Bad Request	Retry on next user invocation/ after restart of client

Scenario	GET/ POST Response Code from ECS	3rd Party Server Action
Invalid or expired Temporary Token in Request	**401** Unauthorized	If possible, trigger device to acquire a (new) valid temporary token from the ECS
Invalid operation in combination with temporary token	**403** Forbidden	Retry on next user invocation/ after restart of client
Requested resource not found	**404** Not found	Retry on next user invocation/ after restart of client
ECS runs into an internal error during procession of request	**500** Internal Server Error	Retry on next user invocation/ after restart of client

Table 8. Error scenarios for Temporary Tokens

2.8.7 Temporary Token Renewal

The ECS should provide the TS.43 client an auth token (e.g., AuthToken, TemporaryToken) with a sufficient time. However, certain ODSA operations (e.g., Subscription Transfer with delayed operation) may take longer than the expected time to complete the operation due to a network error (e.g., between old and new device, to the push server) or the delayed service preparation from the MNO-backend.

When an auth token is expired, the ODSA client and ECS require to start over a procedure from the beginning instead of resuming the procedure, which is inefficient. In addition, if an auth token is expired after the eSIM profile used for the SIM-based authentication has been deleted already, the user has to rely on the Open ID/OAuth authentication that requires user interaction (e.g., create user id and password).

Thus, the ECS/MNO may allow the renewal of a Temporary Token derived from an AuthToken as follows:

- Primary ODSA client requests to an ECS an `AcquireTemporaryToken` operation that has `operation_targets` parameter including AcquireTemporaryToken value.
- If the ECS supports the renewal of a Temporary Token, the ECS returns to the ODSA client `OperationTargets` that includes AcquireTemporaryToken.
- Later, the ODSA client sends an `AcquireTemporaryToken` operation before the time indicated in `TemporaryTokenExpiry`.
- If the ECS receives the `AcquireTemporaryToken` operation request with a Temporary Token as an auth token, then the ECS checks whether it has provided `OperationTargets` with the AcquireTemporaryToken value and the Temporary Token is valid.
- If the ECS has provided the value and the Temporary Token is valid, then the ECS returns a new Temporary Token. It can be the same Temporary Token with a new expiry date. Else otherwise, the ECS shall reject the request.

The number of Temporary Token renewal shall be limited to once.

The following Figure 8a presents how this Temporary Token Renewal fits into the typical steps involved with ODSA entitlement configuration. The steps are:

1. The Old Primary ODSA client sends a request to the ECS to trigger authentication procedure. The ECS and the Old Primary device go through the authentication exchange procedure, and the ECS returns an Auth Token to the ODSA client as a result.
2. The Old Primary ODSA client sends a CheckEligibility request to the ECS. The Old Primary ODSA client and the Old Primary device go through the CheckEligibility procedure. the ECS returns a proper response with application status (ENABLED) as a result.
3. The Old Primary ODSA client sends a request to acquire a temp token to the ECS.
4. The ECS checks the validity of the current Auth Token. If the token is valid and the `operation_targets` includes AcquireTemporaryToken, the ECS may return OperationTargets that includes “AcquireTemporaryToken”.
5. The Old Primary device shall transmit to the New Primary device all relevant information including the Temporary Token information received in step 4 to continue the target ODSA operation(s). The mechanisms to achieve this are outside the scope of this specification.
6. The New Primary ODSA client requests an ODSA operation(s) to the ECS using the Temporary Token. The allowed ODSA operation(s) are listed in the `OperationTargets` returned by the ECS in step 4.
7. If more time is needed to finish the ODSA operation(s), the New Primary ODSA client sends an AcquireTemporaryToken before the time indicated in the TemporaryTokenExpiry.
8. The ECS checks the validity of the current auth token. If the token is valid and the ECS provided OperationTargets including “AcquireTemporaryToken” in step 4, the ECS returns a new Temporary Token or the current Temporary Token with renewed TemporaryTokenExpiry. Otherwise, the ECS rejects the request. The OperationTargets value shall be limited to the value sent in the step 4, but excludes AcquireTemporaryToken.
9. The New Primary ODSA client and the ECS complete the remaining ODSA operation(s) using the Temporary Token received in step 8.

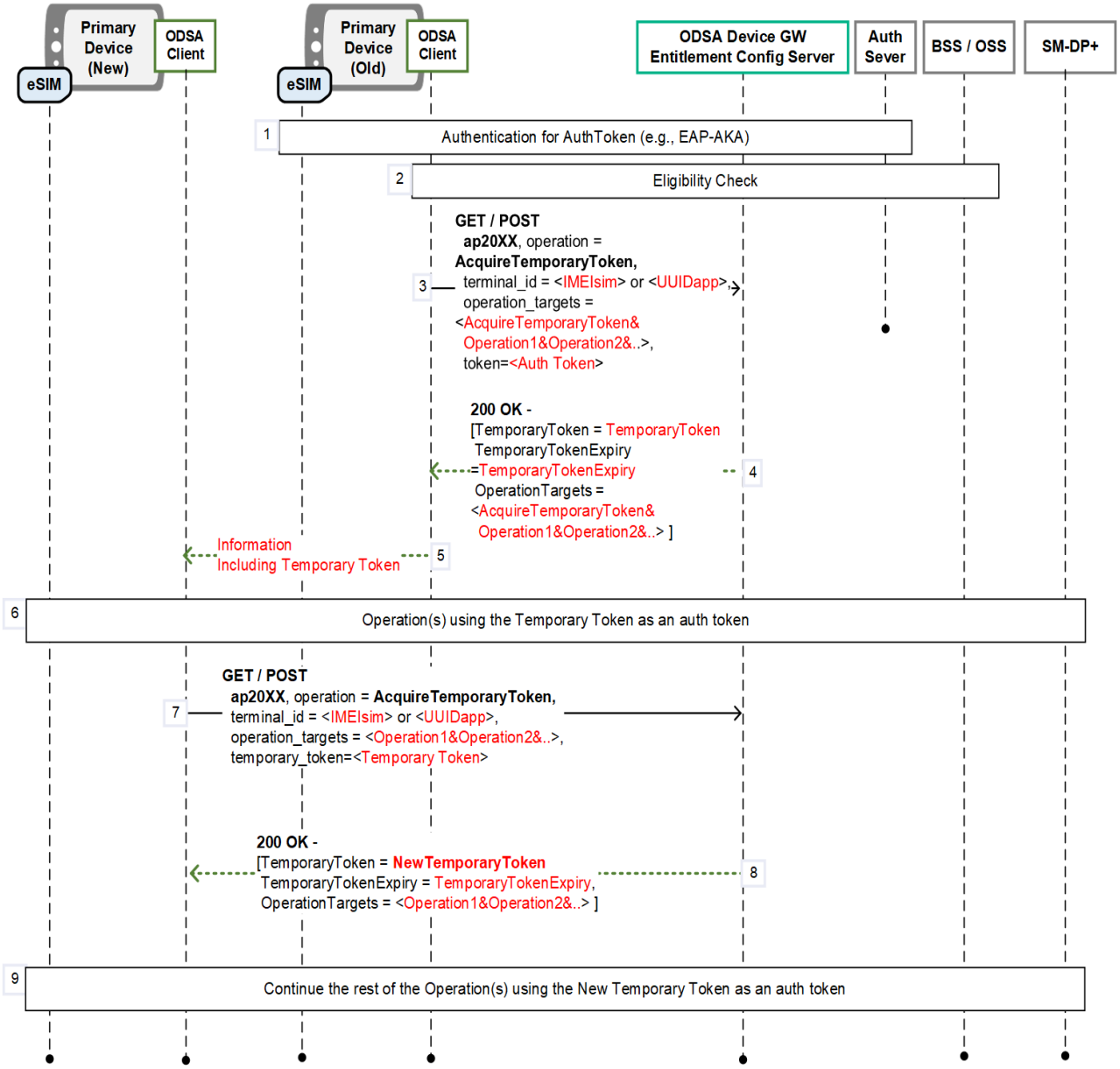

Figure 8a – Temporary Token Renewal Call Flow with Entitlement Configuration Server

2.9 Configuration Document for Entitlements

2.9.1 General

The attributes for the entitlement of VoWiFi, Voice-over-Cellular or SMSoIP and the result of operation requests from Companion and Primary ODSA applications are conveyed between the entitlement configuration server and the client via a configuration document. This document is located in the returned 200 OK response message and can follow two formats:

- An XML document similar to the one defined in section 4 of RCC.14 [5], composed of a set of characteristic types, each with a number of parameters.
- A JSON object composed of a number of structured values (a set of fields presented as name-value pairs) corresponding to the characteristic types of the XML document.

The configuration entitlement server may apply a content encoding mechanism supported by the client.

The client should indicate supported content decoding mechanisms via the Accept-Encoding HTTP header as defined in RFC2616 [12]. The server shall in turn indicate the applied content encoding mechanism in the Content-Encoding HTTP header in accordance with RFC2616 [12].

It is recommended that clients and entitlement configuration servers support the encoding format "gzip".

2.9.2 New Characteristics for XML-Based Document

Extending the XML definition from RCC.14 [5], new APPLICATION characteristics are defined for the entitlements of VoWiFi, Voice-over-Cellular, SMSoIP and for the operation results of the Companion and Primary ODSA applications, with a unique Application Identifier (AppID) assigned to each.

Refer to 2.3 for the AppID assigned to the entitlement applications for VoWiFi, Voice-over-Cellular, SMSoIP and to the Companion and Primary ODSA applications.

An example configuration document containing the combined entitlement parameters for the VoWiFi, Voice-over-Cellular and SMSoIP services is shown in Table 9. This is an example and as such non-normative. The example presents all those entitlements, but only the requested service entitlements shall be included in the document (based on the received “app” request parameter).

For the Companion and Primary ODSA applications, refer to 6.6 for the XML document examples defined for each operation of those applications.

```
<characteristic type="APPLICATION">
    <parm name=”AppID” value=”ap2004”/>
    <parm name=”EntitlementStatus” value=”X”/>
    <parm name=”ServiceFlow_URL” value=”X”/>
    <parm name=”ServiceFlow_UserData” value=”X”/>
    <parm name=”MessageForIncompatible” value=”X”/>
    <parm name=”AddrStatus” value=”X”/>
    <parm name=”TC_Status” value=”X”/>
    <parm name=”ProvStatus” value=”X”/>
</characteristic>
<characteristic type="APPLICATION">
    <parm name=”AppID” value=”ap2003”/>
    <characteristic type="VoiceOverCellularEntitleInfo">
        <characteristic type="RATVoiceEntitleInfoDetails">
            <parm name="AccessType" value="1"/> //4G
            <parm name="HomeRoamingNWType" value="1"/> //Home&Roaming network
            <parm name="EntitlementStatus" value="1"/> //Enabled
        </characteristic>
        <characteristic type="RATVoiceEntitleInfoDetails">
            <parm name="AccessType" value="2"/>  //5G
            <parm name="HomeRoamingNWType" value="2"/> //Home network
            <parm name="EntitlementStatus" value="1"/> //Enabled
            <parm name="NetworkVoiceIRATCapablity" value="EPS-Fallback"/>
        </characteristic>
        <characteristic type="RATVoiceEntitleInfoDetails">
            <parm name="AccessType" value="2"/>  //5G
            <parm name="HomeRoamingNWType" value="3"/> //Roaming network
            <parm name="EntitlementStatus" value="2"/> //Incompatible
            <parm name="MessageForIncompatible" value="Z"/>
        </characteristic>
    </characteristic>
</characteristic>
<characteristic type="APPLICATION">
    <parm name=”AppID” value=”ap2005”/>
    <parm name=”EntitlementStatus” value=”X”/>
</characteristic>
```

Table 9. VoWiFi, Voice-over-Cellular and SMSoIP entitlement document structure (non-normative)

2.9.3 Inclusion in the Complete XML document

The complete XML document with combined VoWiFi, Voice-over-Cellular and SMSoIP entitlement configurations is illustrated in Table 10Table 10. This is an example and as such non-normative. The example presents all those entitlements, but only the requested service entitlements shall be included in the document (based on the received “app” request parameter).

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS">
        <parm name="version" value="X"/>
        <parm name="validity" value="Y"/>
    </characteristic>
    <characteristic type="TOKEN">            <!-- This section is OPTIONAL -->
        <parm name="token" value="U"/>
        <parm name="validity" value="V"/>     <!-- Optional parameter -->
    </characteristic>

<!-- Potentially additional optional characteristics such as MSG, User and Access
Control -->
<!-- see [PRD-RCC.14] -->

    <characteristic type="APPLICATION">
        <parm name=”AppID” value=”ap2004”/>
        <parm name=”EntitlementStatus” value=”X”/>
        <parm name=”ServiceFlow_URL” value=”X”/>
        <parm name=”ServiceFlow_ UserData” value=”X”/>
        <parm name=”MessageForIncompatible” value=”X”/>
        <parm name=”AddrStatus” value=”X”/>
        <parm name=”TC_Status” value=”X”/>
        <parm name=”ProvStatus” value=”X”/>
    </characteristic>
    <characteristic type="APPLICATION">
        <parm name=”AppID” value=”ap2003”/>
        <characteristic type="VoiceOverCellularEntitleInfo">
            <characteristic type="RATVoiceEntitleInfoDetails">
                <parm name="AccessType" value="1"/> //4G
                <parm name="HomeRoamingNWType" value="1"/> //Home&Roaming
                <parm name="EntitlementStatus" value="1"/> //Enabled
            </characteristic>
            <characteristic type="RATVoiceEntitleInfoDetails">
                <parm name="AccessType" value="2"/>  //5G
                <parm name="HomeRoamingNWType" value="2"/> //Home network
                <parm name="EntitlementStatus" value="1"/> //Enabled
                <parm name="NetworkVoiceIRATCapablity" value="EPS-Fallback"/>
             </characteristic>
            <characteristic type="RATVoiceEntitleInfoDetails">
                <parm name="AccessType" value="2"/>  //5G
                <parm name="HomeRoamingNWType" value="3"/> //Roaming network
                <parm name="EntitlementStatus" value="2"/> //Incompatible
                <parm name="MessageForIncompatible" value="Z"/>
            </characteristic>
        </characteristic>
    </characteristic>
    <characteristic type="APPLICATION">
        <parm name=”AppID” value=”ap2005”/>
        <parm name=”EntitlementStatus” value=”X”/>
    </characteristic>
</wap-provisioningdoc>
```

Table 10. Complete XML-based entitlement document structure (non-normative)

2.9.4 JSON-Based Configuration Document

The JSON object value returned as part of an entitlement configuration request for the entitlements of VoWiFi, Voice-over-Cellular and SMSoIP is presented in Table 11. Each characteristic type of the XML document is mapped to the JSON document as a structured object with several fields.

For the Companion and Primary ODSA applications, refer to 6.6 for a description of the JSON-based document defined for each operation of those applications.

```
{
    "Vers" : {
        "version" : "X",
        "validity" : "Y"
    },
    "Token" : {              // Optional
        "token" : "U",
        "validity" : "V"
    },
    "ap2004": {            // VoWiFi Entitlement settings
        "EntitlementStatus" : "X",
        "ServiceFlow_URL" : "X",
        "ServiceFlow_ UserData" : "X",
        "MessageForIncompatible" : "X",
        "AddrStatus" : "X",
        "TC_Status" : "X",
        "ProvStatus" : "X"
    },
    "ap2003" : {            // Voice-over-Cellular Entitlement settings
        "VoiceOverCellularEntitleInfo" : [{
             "RATVoiceEntitleInfoDetails" : {
                "AccessType" : "1", //4G
                "HomeRoamingNWType" : "1", //Home & Roaming networks
                "EntitlementStatus" : "1"  //Enabled
             }
           },{
              "RATVoiceEntitleInfoDetails" : {
                "AccessType" : "2", //5G
                "HomeRoamingNWType" : "2", //Home Network
                "EntitlementStatus" : "1", //Enabled
                "NetworkVoiceIRATCapablity" : "EPS-Fallback"
             }
           },{
              "RATVoiceEntitleInfoDetails" : {
                "AccessType" : "2", //5G
                "HomeRoamingNWType" : "3", //Roaming Network
                "EntitlementStatus" : "2", //Incompatible
                "MessageForIncompatible" : "Z"
             }
        }]
    },
        "ap2005" : {           // SMSoIP Entitlement settings
        "EntitlementStatus" : "X"
    }
}
```

Table 11. JSON-based entitlement document for VoWiFi, Voice-over-Cellular and SMSoIP (non-normative)

2.9.5 Result of Notification Registration

An application can request to receive entitlement notifications from the network by including the `notif_action` and `notif_token` parameters in a configuration request (refer to Table 4 for details on the parameters).

The Entitlement Configuration Server shall provide the result of registering the application in the configuration document using the `RegisterNotifStatus` configuration parameter as defined in Table 12.

General Entitlement parameter	Type	Values	Description
RegisterNotifStatus (Conditional)	Integer	0 - SUCCESS	Registration of the notification was successful
		1 – INVALID TOKEN	The provided notif_token was invalid
		2 – DUPLICATE TOKEN	The provided notif_token is a duplicate

Table 12. Entitlement Parameter - Notification Registration Status

2.9.6 Additional Details on TOKEN

As seen in Table 10 and Table 11, the document for entitlement configuration contains the VERS and TOKEN attributes, as defined by RCC.14 [5]. In addition to the definition of TOKEN from RCC.14, the following rules apply to the entitlement configuration's TOKEN:

- TOKEN is not restricted to entitlement configuration requests made from non-3GPP access networks access types.
- A "`validity`" attribute is allowed and indicates the lifetime of the provided token.
- The token shall be kept by clients during reboot cycles.
- The token is of variable length.

2.10 HTTP Response Codes

Table 13 presents the possible entitlement configuration server response codes (including associated reasons) at the HTTP level.

GET Response Code	Reason	Device's Action
200 OK + with application data	New or updated application data sent to the device, including ODSA responses with error indication `OperationResult!=0`	Process the returned application data
302 Found	OAuth 2.0 / OpenID Connect authentication should be followed. Refer to Section 2.8.2 for details on the procedure and its initiation.	Redirect the GET request to the OIDC AuthN endpoint specified by the **Location**: field of the 302 Found response
400 Bad Request	Invalid or missing GET parameters or wrong format	Retry on next reboot/the next time the client app starts
403 Forbidden	Invalid identities (device id, primary or companion) or the operation is supported but is not allowed by the ECS for this `requestor_id`.	Retry on next reboot/the next time the client app starts
405 Method not Allowed	Operation is known by the server but is not supported.	Retry on next reboot/the next time the client app starts.

GET Response Code	Reason	Device's Action
406 Not Acceptable	The server does not support the `entitlement_version` used by the client, or the server doesn't support device transfer functionality using `old_companion _terminal_iccid` and `old_companion _terminal_id`	Apply the procedure defined by the Service Provider for the case of no configuration data is available (for example silent abort or error message)
500 Internal Server error	Internal error during processing of GET request	Retry on next reboot/the next time the client app starts
501 Not implemented	The server does not support the HTTP POST method used by the client	Retry the request using GET method
503 Retry after / Service Unavailable	The server does not have access to external resources (temporary error)	Retry after the time specified in the "**Retry-After**" header
511 Network Authentication Required	To initiate authentication with the server, when proper AuthN parameters are missing, the `OTP` is invalid, or the `token` obtained through a previous authentication exercise expired	Client app should go through an authentication procedure with the entitlement configuration server and get a new `token` **Alternate Authentication fall-back**: If the client fails to obtain a new `token` using EAP-AKA authentication (the EAP_ID parameter present) and receives a 511 response it shall initiate authentication with the ECS without including the EAP_ID parameter.
The server is unreachable	Entitlement configuration server is missing or down	Retry on next reboot, the next time the client starts

Table 13. HTTP Response Codes from Entitlement Configuration Server

3 VoWiFi Entitlement Configuration

The following sections describe the different configuration parameters associated with the VoWiFi entitlement as well as the expected behaviour of the VoWiFi client based on the entitlement configuration document received by the client.

3.1 VoWiFi Entitlement Parameters

Parameters for the VoWiFi entitlement provide the overall status of the VoWiFi service to the client, as well as the different sub-status associated with the activation procedure of the service.

The VoWiFi entitlement parameters also include information associated with the web views presented to users by the VoWiFi client during activation and management of the service.

3.1.1 VoWiFi Entitlement Status

- Parameter Name: `EntitlementStatus`
- Presence: Mandatory

This parameter indicates the overall status of the VoWiFi entitlement, stating if the service can be offered on the device, and if it can be activated or not by the end-user.

The different values for the VoWiFi entitlement status are provided in Table 14.

VoWiFi Entitlement parameter	Type	Values	Description
EntitlementStatus (Mandatory)	Integer	0 - DISABLED	VoWiFi service allowed, but not yet provisioned and activated on the network side
		1 - ENABLED	VoWiFi service allowed, provisioned, and activated on the network side
		2 - INCOMPATIBLE	VoWiFi service cannot be offered
		3 - PROVISIONING	VoWiFi service being provisioned on the network side

Table 14. Entitlement Parameter - VoWiFi Overall Status

3.1.2 VoWiFi Client's Web Views Parameters

- Parameter Names: `ServiceFlow_URL` and `ServiceFlow_UserData`
- Presence: Mandatory

During the activation procedure of the VoWiFi service, end-users can be presented with web views specific to the Service Provider. VoWiFi web views allow end-users to change user-specific attributes of the VoWiFi service, like the acceptance of the service's Terms and Conditions (T&C) and the end-user's physical address (needed in some regions for VoWiFi emergency calling purposes).

The entitlement parameters associated with the VoWiFi service's web views are described in Table 15.

VoWiFi Entitlement parameter	Type	Description
ServiceFlow_URL (Mandatory)	String	The URL of web views to be used by VoWiFi client to present the user with VoWiFi service activation and service management options, which may include entering physical address and agreeing to the T&C of the VoWiFi service.
ServiceFlow_UserData (Mandatory)	String	User data associated with the HTTP web request towards the ServiceFlow URL. It can contain user-specific attributes to ease the flow of VoWiFi service activation and management. See below for details on the content.

Table 15. Entitlement Parameters - VoWiFi Web Views Information

The content of the `ServiceFlow_UserData` parameter is defined by the requirements of the Service Provider's VoWiFi web views. In a typical case, the web view is presented when VoWiFi service is activated by the end-user. At such time the VoWiFi client connects the user to the `ServiceFlow_URL` and includes the `ServiceFlow_UserData` in the HTTP web request.

In order to improve user experience, this parameter should include user and service-specific information that would allow the VoWiFi's web views to identify the requestor and be aware of the latest VoWiFi entitlement status values.

An example of the `ServiceFlow_UserData` string is:

```
"imsi=XXXXXXXXX&amp;msisdn=XXXXXXXX&amp;tnc=X&ampaddr=X&amp;prov=X&amp;device_id=XXXXXXXX&amp;entitlement_name=VoWiFi&amp;signature=Xl%2F1tT23C0dNI32hiVZZS"
```

This example contains elements associated with the device and user identities as well as service-related information like the current T&C, address, and provisioning status of the VoWiFi service. Note the use of "`&`" is required to allow the '&' character to be used in a string value within an XML document.

3.1.3 VoWiFi Address Parameters

- Parameter Name: `AddrStatus, AddrExpiry, AddrIdentifier`
- Presence:
  - `AddrStatus`: Mandatory
  - `AddrExpiry`, `AddrIdentifier`: Optional

In some regions, end-users must provide their static physical address before being allowed to use the VoWiFi service. Those entitlement parameters indicates if that condition must be met before offering the VoWiFi service and provide additional information on the captured location (expiration and identifier).

Also, if a physical address from the end-user is indeed needed for the VoWiFi service, this parameter indicates the state of the “address capture” process.

The different values for the VoWiFi address status are provided in Table 16.

VoWiFi Entitlement parameter	Type	Values	Description
AddrStatus (Mandatory)	Integer	0 - NOT AVAILABLE	Address has not yet been captured from the end-user
		1 - AVAILABLE	Address has been entered by the end-user
		2 - NOT REQUIRED	Address is not required to offer VoWiFi service
		3 - IN PROGRESS	Address capture from end-user is on-going
AddrExpiry (Optional)	Time	in ISO 8601 format, of the form YYYY-MM-DDThh:mm:ssTZD	The time/date when the address expires and should be recaptured from the user
AddrIdentifier (Optional)	String	Generated by emergency system	Associated identifier of the location, to be used during an IMS emergency session by the device, as defined in 3.1.3.

Table 16. Entitlement Parameters - VoWiFi Address

The absence of the `AddrExpiry` parameter indicates that there is no expiration date for the address.

3.1.4 VoWiFi T&C Status

- Parameter Name: `TC_Status`
- Presence: Mandatory

In some regions, end-users must agree to the Terms and Conditions (T&C) of the VoWiFi service before being allowed to use it. This entitlement parameter indicates if that condition must be met before offering the VoWiFi service.

Also, if acceptance of the VoWiFi’s T&C is indeed needed from the end-user, this parameter indicates the state of the “T&C acceptance” process.

The different values for the VoWiFi T&C status are provided in Table 17.

VoWiFi Entitlement parameter	Type	Values	Description
TC_Status (Mandatory)	Integer	0 - NOT AVAILABLE	T&C have not yet been accepted by the end-user
		1 - AVAILABLE	T&C have been accepted by the end-user
		2 - NOT REQUIRED	T&C acceptance is not required to offer VoWiFi service

VoWiFi Entitlement parameter	Type	Values	Description
		3 - IN PROGRESS	T&C capture and acceptance is on-going

Table 17. Entitlement Parameter - VoWiFi T&C Status

3.1.5 VoWiFi Provisioning Status

- Parameter Name: `ProvStatus`
- Presence: Mandatory

In some cases, the network is not provisioned by default to support VoWiFi service for all end-users. Some type of network-side provisioning must then take place before offering the VoWiFi service to the end-user. This entitlement parameter indicates the progress of VoWiFi provisioning on the network for the requesting client.

The different values for the VoWiFi provisioning status are provided in Table 18.

VoWiFi Entitlement parameter	Type	Values	Description
ProvStatus (Mandatory)	Integer	0 - NOT PROVISIONED	VoWiFi service not provisioned yet on network side
		1 - PROVISIONED	VoWiFi service fully provisioned on network
		2 - NOT REQUIRED	Provisioning progress of VoWiFi is not tracked / not required
		3 - IN PROGRESS	VoWiFi provisioning is still in progress

Table 18. Entitlement Parameter - VoWiFi Provisioning Status

3.1.6 VoWiFi Message for Incompatible Status

- Parameter Name: `MessageForIncompatible`
- Presence: Mandatory

When the status for the VoWiFi entitlement is INCOMPATIBLE (see 3.1.1) and the end-user tries to activate VoWiFi, the VoWiFi client should show a message to the end-user indicating why activation was refused.

This entitlement parameter provides the content of that message, as decided by the Service Provider. Table 19 describes this VoWiFi entitlement parameter.

VoWiFi Entitlement parameter	Type	Description
MessageForIncompatible (Mandatory)	String	A message to be displayed to the end-user when activation fails due to an incompatible VoWiFi Entitlement Status

Table 19. Entitlement Parameter - VoWiFi Message for Incompatible Status

3.2 Client Behaviour for VoWiFi Entitlement Configuration

The entitlement parameters for VoWiFi provides an overall status for the service as well as additional information associated with the activation procedure and provisioning of the service.

As such, the entitlement configuration for VoWiFi carries information that impacts the behaviour of the VoWiFi client.

The client shall then activate (or deactivate) the VoWiFi service according to the combination of the VoWiFi's general setting on the device (controlled by the end-user) and the received VoWiFi entitlement configuration.

The client shall also use the VoWiFi entitlement parameters to decide if VoWiFi web views for activation and service management should be presented to the end-user. This includes country-specific details on the need for VoWiFi's Terms & Conditions acceptance and the requirement to capture or not the user's physical address - a country's regulations may require users to enter their physical address as well as agree to the Terms & Conditions of the service when activating VoWiFi.

3.3 Entitlement Modes of VoWiFi Client

To simplify the description of the client's behaviour with respect to the VoWiFi entitlement configuration, a set of "VoWiFi entitlement modes" for the client is defined, each with specific expectations on the client side.

The relationship between the values of the VoWiFi entitlement parameters and the VoWiFi entitlement modes are shown in Table 20.

VoWiFi Entitlement Parameters				**VoWiFi Entitlement mode**
Entitlement Status	**ProvStatus**	**TC_Status**	**AddrStatus**	
INCOMPATIBLE	Any			Cannot be offered
DISABLED	Any	At least one is NOT AVAILABLE		Service Data Missing
		At least one is IN PROGRESS		Service Data being Updated
DISABLED	NOT PROVISIONED, IN PROGRESS	AVAILABLE or NOT REQUIRED		Service being Provisioned
PROVISIONING	Any			
ENABLED	PROVISIONED or NOT REQUIRED	AVAILABLE or NOT REQUIRED		Can be activated

Table 20. VoWiFi Entitlement Modes

The description of each VoWiFi entitlement mode follows.

3.3.1 VoWiFi Entitlement Mode - Cannot be offered.

The Client shall stay in this mode when:

- **`EntitlementStatus`** is INCOMPATIBLE

The Client shall not activate the VoWiFi service.

Due to end-user's action, the client may send a request to the Entitlement Configuration Server to refresh the VoWiFi entitlement status. If the received status is still INCOMPATIBLE, the device shall either display **`MessageForIncompatible`** when it is not void, or the default device error message (if any).

3.3.2 VoWiFi Entitlement Mode - Can be activated.

The Client shall stay in this mode when all the following conditions are met:

- **`EntitlementStatus`** is ENABLED
- **`ProvStatus`** is PROVISIONED or NOT REQUIRED
- **`TC_status`** and **`AddrStatus`** are AVAILABLE or NOT REQUIRED

When entering this mode, the client shall activate the VoWiFi service if the VoWiFi's service setting on the device is equivalent to ON (may require end-user action).

3.3.3 VoWiFi Entitlement Mode - Service Data Missing

The Client shall stay in this mode when all the following conditions are met:

- **`EntitlementStatus`** is DISABLED
- **`ProvStatus`** is any values.
- Either **`TC_status`** or **`AddrStatus`** is NOT AVAILABLE

In that mode the Client shall not activate the VoWiFi service.

Due to end-user's action, the Client may send a request to the Entitlement Configuration Server to refresh the VoWiFi entitlement status. If the received status leads to the same mode, the Client shall open a web view and instruct the end-user to enter the required missing VoWiFi service information (T&C or static physical address).

3.3.4 VoWiFi Entitlement Mode - Service Data Being Updated

The Client shall stay in this mode when all the following conditions are met:

- **`EntitlementStatus`** is DISABLED
- **`ProvStatus`** is any values.
- Either **`TC_status`**, or **`AddrStatus`** is set to IN PROGRESS

In that mode the Client shall not activate the VoWiFi service.

3.3.5 VoWiFi Entitlement Mode - Service Being Provisioned

The Client shall stay in this mode when all the following conditions are met:

- **EntitlementStatus** is DISABLED
- **TC_status** and **AddrStatus** are set to AVAILABLE or NOT REQUIRED
- **ProvStatus** is set to NOT PROVISIONED or IN PROGRESS

Or

- **EntitlementStatus** is PROVISIONING
- **ProvStatus**, **TC_status** and **AddrStatus** are set to any values.

The Client shall not activate the VoWiFi service. After an end-user action (going into VoWiFi service settings for example), the client shall show that the service is pending or being provisioned.

3.4 VoWiFi Client Considerations around Web View Callbacks

During the activation procedure of the VoWiFi service, end-users can be presented with web views specific to the Service Provider (hosted by a VoWiFi portal web server). To support this feature, the VoWiFi entitlement parameters **ServiceFlow_URL** and **ServiceFlow_UserData** associated with the invocation of VoWiFi service's web views by the VoWiFi client are defined in section 3.1.2.

At the completion of the web service flow by the VoWiFi portal web server, the web page shall invoke a specific JavaScript (JS) callback function associated with the VoWiFi client. The callback functions shall provide the overall state of the web flow to the VoWiFi client and indicate that the VoWiFi web view on the device needs to be closed.

The object associated with the callback functions is **VoWiFiWebServiceFlow** and two different callback functions are defined to reflect the state of the web logic.

3.4.1 entitlementChanged() Callback function

The **entitlementChanged()** callback function indicates that the VoWiFi service flow ended properly between the device and VoWiFi portal web server.

The web view to the end-user should be closed and the VoWiFi client shall make a request for the latest VoWiFi entitlement configuration status, via the proper TS.43 entitlement configuration request.

Based on the returned set of status parameters, the VoWiFi client shall behave as specified in 3.3.

The following call flow presents how the **entitlementChanged()** callback function fits into the typical steps involved with VoWiFi entitlement configuration. At the end of the VoWiFi service flow the callback function (step 7) is invoked by the web server and the VoWiFi client acts accordingly by requesting for the latest VoWiFi entitlement configuration.

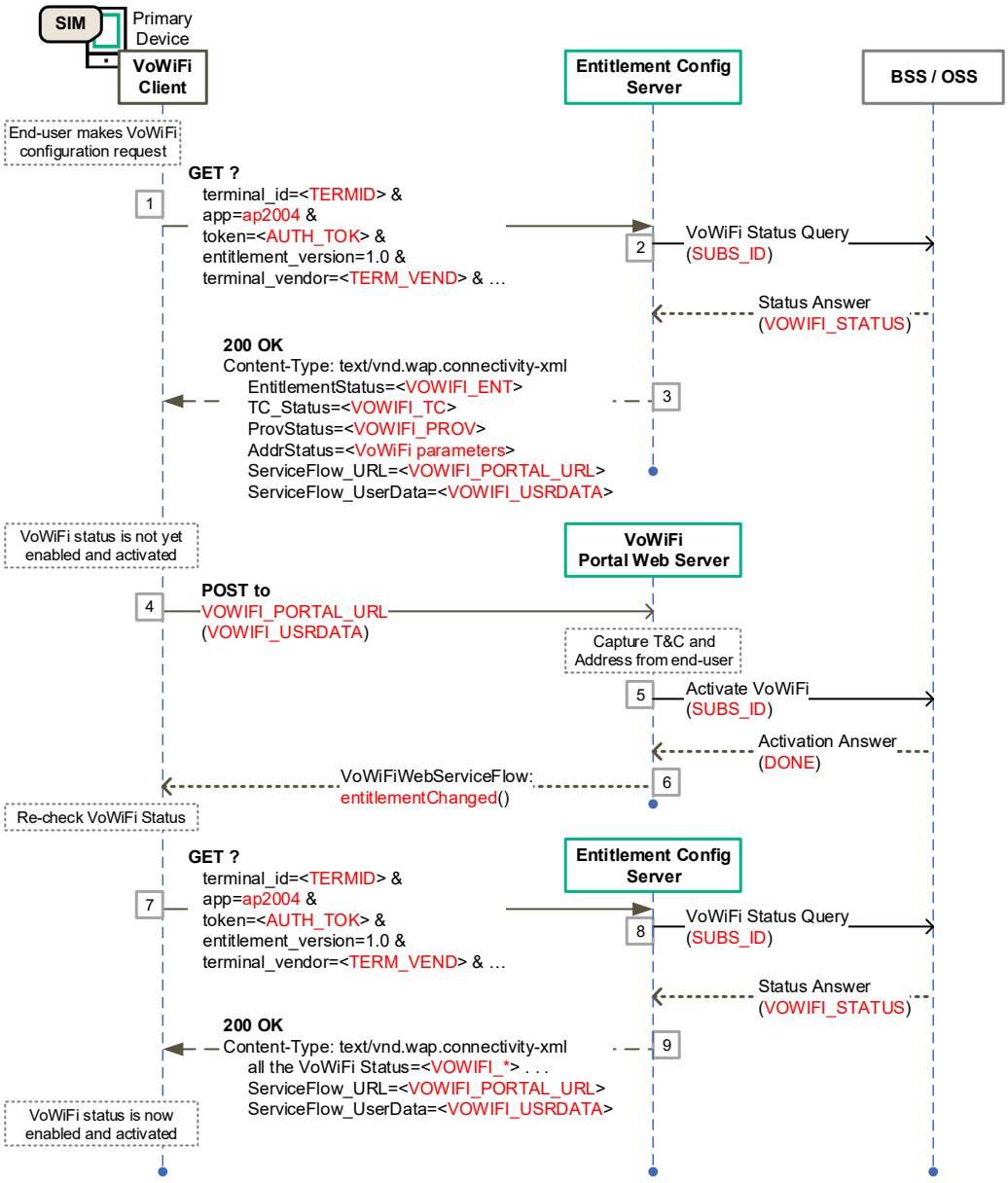

Figure 8. VoWiFi Entitlement Configuration Flow with entitlementChanged() Callback

3.4.2 dismissFlow() callback function

The **`dismissFlow()`** callback function indicates that the VoWiFi service flow ends prematurely, either caused by user action (DISMISS button for example) or by an error in the web sheet logic or from the network side.

As a result of the dismissal of the service flow, the VoWiFi entitlement status has not been updated by the VoWiFi portal.

The web view to the end-user should be closed and the VoWiFi client should not make a request for the latest VoWiFi entitlement configuration status.

The call flow in Figure 9 presents how the **`dismissFlow()`** callback function fits into the typical steps involved with VoWiFi Entitlement Configuration. Due to an error or user action

the callback function (step 6) is invoked by the web server and the VoWiFi client acts accordingly.

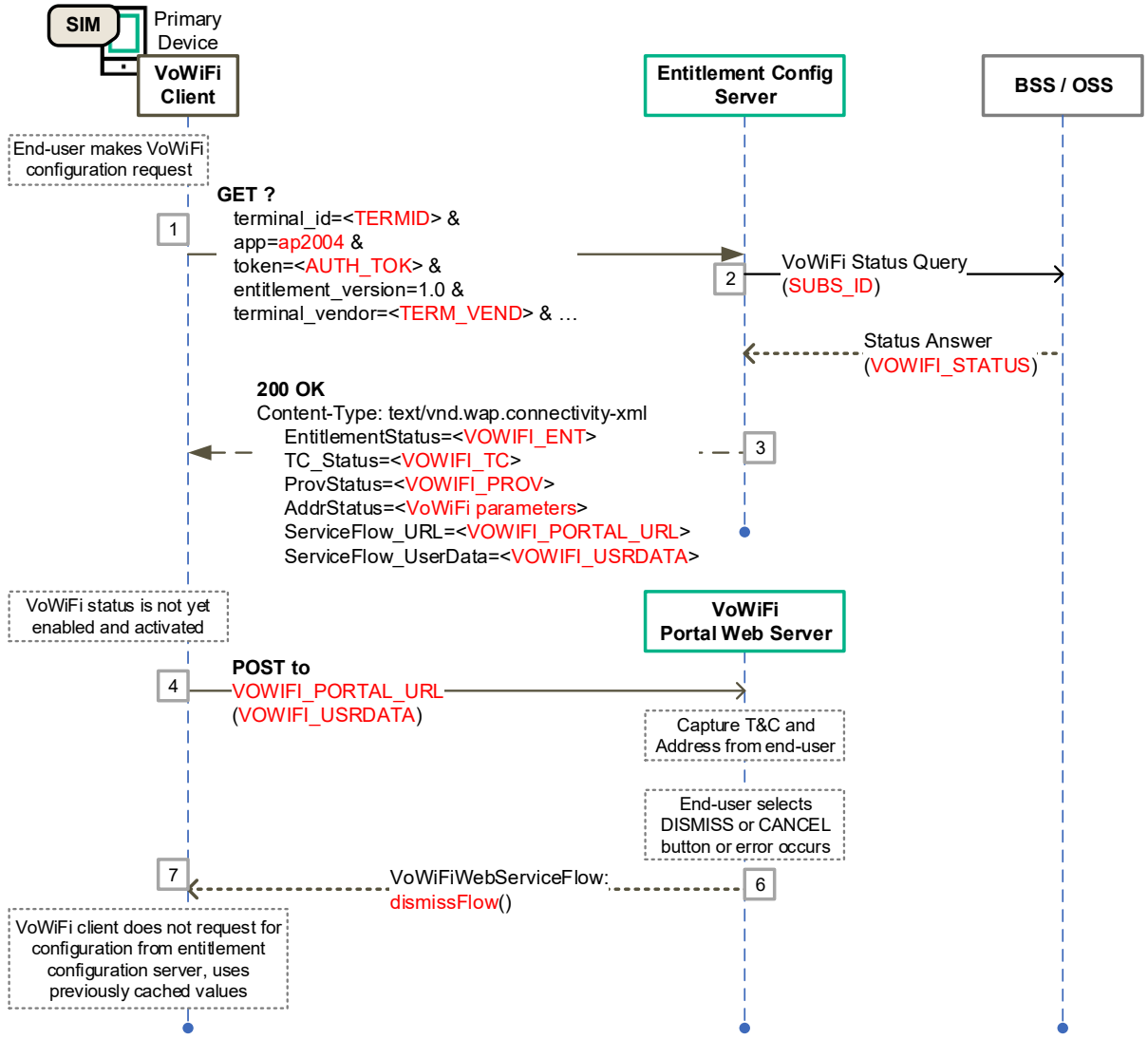

Figure 9. VoWiFi Entitlement Configuration Flow with dismissFlow() Callback

4 Voice-over-Cellular Entitlement Configuration

The following sections describe the different configuration parameters associated with the Voice-over-Cellular entitlement.

Note: For TS.43 version earlier than 7.0, AppID “ap2003” is only used for VoLTE entitlement. If Entitlement Configuration Server or device plans to support VoLTE entitlement only and use version 6.0 or earlier, please refer to the section 4, *VoLTE Entitlement Configuration,* of TS.43 version 6.0 or earlier. From TS.43 version 7.0 onwards, AppID “ap2003” is extended to be used for Voice-over-Cellular entitlement configuration for different cellular Radio Access Types (RATs). That is, after a device passes EAP-AKA authentication with Entitlement Configuration Sever, Entitlement Configuration Server can share voice configuration parameters of different cellular RATs to the device, such as 4G VoLTE and 5G Voice over New Radio (VoNR) entitlement configurations.

4.1 Voice-over-Cellular Entitlement Parameters

Parameters for Voice-over-Cellular entitlement provide the overall voice entitlement status of different cellular RATs to the device or client.

4.1.1 Voice-over-Cellular Entitlement Parameter Definition

The following 2 parameters are defined for Voice over Cellular Entitlement:

- **`VoiceOverCellularEntitleInfo`:** Top level, list of cellular voice entitlement info associated with the device's client.
- **`RATVoiceEntitleInfoDetails`**: Each `RATVoiceEntitleInfoDetails` provides the voice entitlement parameters for a specific RAT in home and/or roaming conditions. Within `VoiceOverCellularEntitleInfo`, it can have one or more `RATVoiceEntitleInfoDetails` parameters for each supported RAT type.

`RATVoiceEntitleInfoDetails` is a multi-parameter structure. The `RATVoiceEntitleInfoDetails` structure has the parameters listed in Table 21 below.

“RATVoiceEntiltmenetInfoDetails” configuration parameters	Type	Value	Description
AccessType (mandatory)	Integer	1 - LTE	RAT of type LTE (4G)
		2 – 5G NG-RAN	RAT of type NG-RAN (5G)
HomeRoamingNWType (mandatory)	Integer	1 - All (include both home and roaming networks)	Voice service entitlement configurations for both home and roaming networks.
		2 – Home network type	Voice service entitlement configurations for home network.
		3 -Roaming network type	Voice service entitlement configurations for roaming network.

“RATVoiceEntiltmenetInfoDetails” configuration parameters	Type	Value	Description
EntitlementStatus (mandatory)	Integer	0 - DISABLED	Voice service allowed, but not yet provisioned and activated on the network.
		1 - ENABLED	Voice service allowed, provisioned, and activated on the network
		2 - INCOMPATIBLE	Voice service cannot be offered for network
		3 - PROVISIONING	Voice service being provisioned on the network
MessageForIncompatible (conditional)	String	The content of the message is decided by the Service Provider.	A message to be displayed to the end-user when activation fails due to an incompatible voice Entitlement Status for this RAT. When the status for the voice entitlement is INCOMPATIBLE and the end-user tries to activate voice entitlement for this RAT, the client should show a message to the end-user indicating why activation was refused. This parameter is defined as conditional type, which means Entitlement Configuration Server sends `MessageForIncompatible` parameter to device only when its `EntitlementStatus` is INCOMPATIBLE.

"RATVoiceEntiltmenetInfoDetails" configuration parameters	Type	Value	Description
NetworkVoiceIRATCapability (optional)	String	One of the following defined string values for a given RAT • "EPS-Fallback" (5G only) • "5G-SRVCC" (5G only) • "4G-SRVCC" (4G only)	`NetworkVoiceIRATCapability` can be used by network to share network supported Inter-RAT voice service capabilities to device for a given RAT. An example of `NetworkVoiceIRATCapability` for 5G RAT is shown as below: • "EPS-Fallback" It means 5G network supports EPS Fallback for voice call. Entitlement Configuration Server shall include this optional parameter when EPS-Fallback is the only possible procedure for voice services i.e. UE will perform a fallback from 5G/NR to 4G/LTE in order to establish a voice call.

Table 21: RATVoiceEntitleInfoDetails - Cellular Voice Entitlement Details of a Given RAT

4.1.2 Voice-over-Cellular Entitlement Response Example

Table 22 represents an example for a returned Voice-over-Cellular entitlement configuration in XML format for VoLTE and VoNR.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS">
         <parm name="version" value="X"/>
         <parm name="validity" value="Y"/>
     </characteristic>

     <characteristic type="TOKEN">
         <parm name="token" value="U"/>
     </characteristic>

     <characteristic type="APPLICATION">
        <parm name="AppID" value="ap2003"/>
        <characteristic type="VoiceOverCellularEntitleInfo">
            <characteristic type="RATVoiceEntitleInfoDetails">
                <parm name="AccessType" value="1"/>  //4G
                <parm name="HomeRoamingNWType" value="1"/> //Home&Roaming
                <parm name="EntitlementStatus" value="1"/>  //Enabled
            </characteristic>
            <characteristic type="RATVoiceEntitleInfoDetails">
                <parm name="AccessType" value="2"/>  //5G
                <parm name="HomeRoamingNWType" value="2"/> //Home network
                <parm name="EntitlementStatus" value="1"/> //Enabled
                <parm name="NetworkVoiceIRATCapablity" value="EPS-Fallback"/>
            </characteristic>
            <characteristic type="RATVoiceEntitleInfoDetails">
                <parm name="AccessType" value="2"/>  //5G
                <parm name="HomeRoamingNWType" value="3"/> //Roaming network
                <parm name="EntitlementStatus" value="2"/> //Incompatible
                <parm name="MessageForIncompatible" value="Z"/>
            </characteristic>
        </characteristic>
     </characteristic>
</wap-provisioningdoc>
```

Table 22: Example of Voice over Cellular Entitlement response in XML format

Table 23 represents an example for a returned Voice-over-Cellular entitlement configuration in JSON format for VoLTE and VoNR.

```
{
    "Vers" : {
        "version" : "X",
        "validity" : "Y"
    },
    "Token" : {              // Optional
        "token" : "U"
    },
    "ap2003" : {            // Voice over Cellular
        "VoiceOverCellularEntitleInfo" : [{
              "RATVoiceEntitleInfoDetails" : {
                "AccessType" : "1", //4G
                "HomeRoamingNWType" : "1",//Home & Roaming network
                "EntitlementStatus" : "1" //Enabled
            }
           },{
              "RATVoiceEntitleInfoDetails" : {
                "AccessType" : "2", //5G
                "HomeRoamingNWType" : "2", //Home Network
                "EntitlementStatus" : "1", //Enabled
                "NetworkVoiceIRATCapablity" : "EPS-Fallback"
            }
           },{
              "RATVoiceEntitleInfoDetails" : {
                "AccessType" : "2", //5G
                "HomeRoamingNWType" : "3", //Roaming Netowrk
                "EntitlementStatus" : "2",//Incompatible
                "MessageForIncompatible" : "Z"
            }
        }]
    }
}
```

Table 23: Example of Voice over Cellular Entitlement response in JSON format

5 SMSoIP Entitlement Configuration

The following sections describe the different configuration parameters associated with the SMSoIP entitlement as well as the expected behaviour of the SMSoIP client based on the entitlement configuration document received by the client.

5.1 SMSoIP Entitlement Parameters

Parameters for the SMSoIP entitlement provide the overall status of the SMSoIP service to the client and other client-related information.

5.1.1 SMSoIP Entitlement Status

- Parameter Name: `EntitlementStatus`
- Presence: Mandatory

This parameter indicates the overall status of the SMSoIP entitlement, stating if the service can be offered on the device, and if it can be activated or not by the end-user.

The different values for the SMSoIP entitlement status are provided in Table 24.

SMSoIP Entitlement parameter	Type	Values	Description
EntitlementStatus (Mandatory)	Integer	0 - DISABLED	SMSoIP service allowed, but not yet provisioned and activated on the network side
		1 - ENABLED	SMSoIP service allowed, provisioned, and activated on the network side
		2 - INCOMPATIBLE	SMSoIP service cannot be offered
		3 - PROVISIONING	SMSoIP service being provisioned on the network side

Table 24. Entitlement Parameter - SMSoIP Overall Status

5.2 Client Behaviour to SMSoIP Entitlement Configuration

The client shall activate (or deactivate) the SMSoIP service according to the combination of the SMSoIP settings on the device (controlled by the end-user) and the received SMSoIP Entitlement status described in this document. This is presented in Table 25

SMSoIP Entitlement Status	SMSoIP Client Behavior
INCOMPATIBLE	The Client shall not activate the SMSoIP service. The client may send a request to the Entitlement Configuration Server to refresh the SMSoIP entitlement status.

SMSoIP Entitlement Status	SMSoIP Client Behavior
DISABLED	The Client shall not activate the SMSoIP service. After an end-user action (going into SMSoIP's service settings for example), the client may send a request to the Entitlement Configuration Server to refresh the SMSoIP entitlement status.
PROVISIONING	The Client shall not activate the SMSoIP service. After an end-user action (going into SMSoIP's service settings for example), the client shall show that the service is pending or being provisioned.
ENABLED	The client shall activate the SMSoIP service if the SMSoIP's service setting on the device is equivalent to ON (may require end-user action).

Table 25. SMSoIP Client Behaviour

6 On-Device Service Activation (ODSA) Entitlement and Configuration

The ODSA procedure for eSIM-based devices is initiated by a client application on a requesting or primary device. The ODSA application requires entitlement and configuration information from the Service Provider in order to complete the procedure. The following sections present the different operations associated with ODSA of eSIM devices and the resulting configuration documents.

6.1 ODSA Architecture and Operations

The ODSA client application runs on a requesting or primary device and allows the end-user to perform a seamless activation of the subscription and associated services on the eSIM of either a companion device or the primary device, without involvement of Service Provider's customer or support personnel.

In order to have access to the eSIM, the ODSA client application shall be invoked at the request of the end-user and shall capture proper interactions (e.g. user consent) as described in SGP.21 [10] and SGP.22 [11].

The architecture for the companion ODSA use case is shown in Figure 10. The Entitlement Configuration Server acts as the Service Provider's ODSA Gateway for the ODSA procedure (labelled as the "ODSA GW" in Figure 10), providing entitlement and configuration data to the "ODSA for Companion devices" application.

The device hosting the ODSA client is the "requesting" device. It may or may not have access to a SIM with an active profile from the Service Provider. The interface between the ODSA client on the requesting device and the companion device is out-of-scope of this specification.

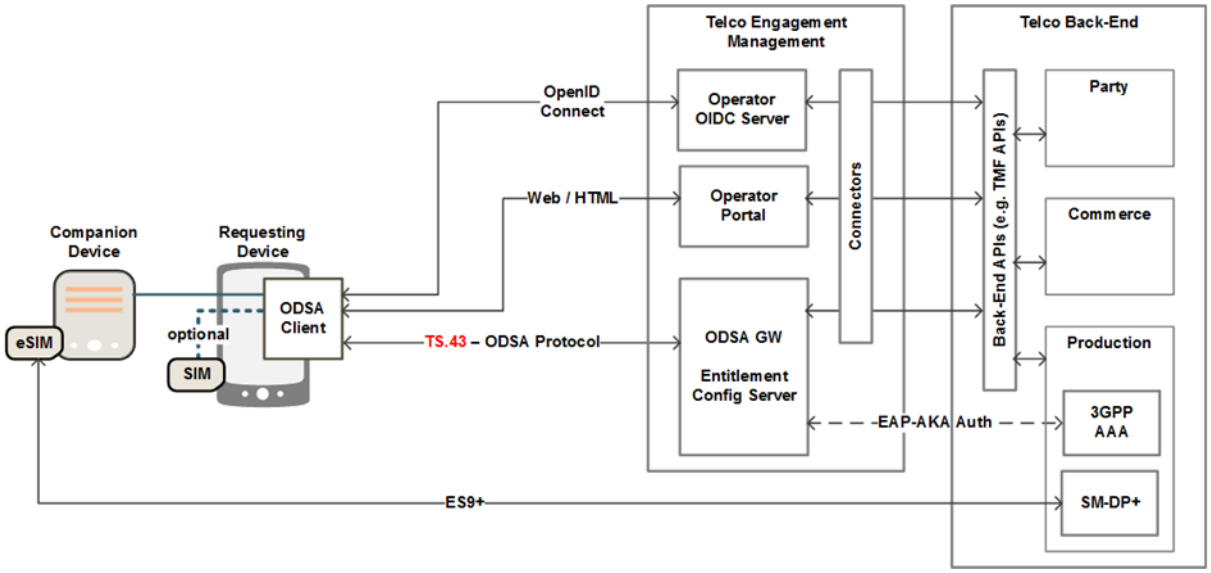

Figure 10. ODSA for Companion eSIM devices, architecture, and TS.43 positioning

The architecture for primary ODSA use case is shown in Figure 11. The device is "primary" as it has direct access to the eSIM being activated through the ODSA procedure. As in the

companion ODSA use case, the ODSA may or may not have access to a SIM with an active profile from the Service Provider. The interface between the ODSA client and the eSIM is out-of-scope of this specification.

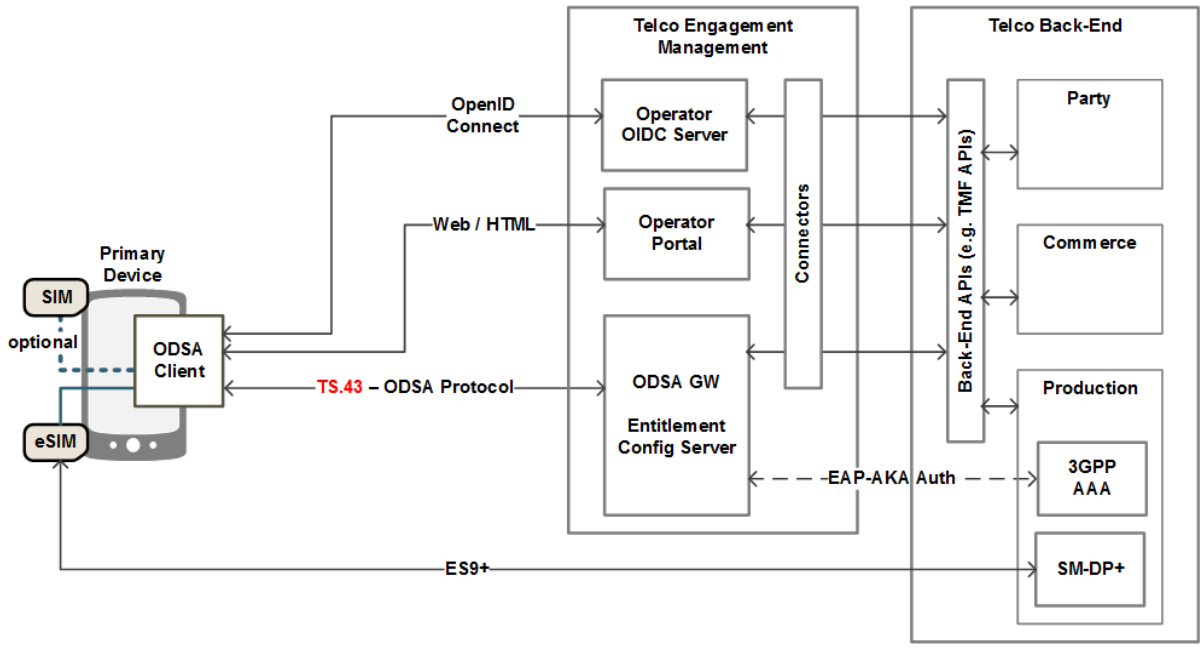

Figure 11. ODSA for Primary eSIM devices, architecture, and TS.43 positioning

This specification does not cover the HTML-based interactions between the ODSA application and the Service Provider's portal web server (labelled as the "Operator Portal" in Figure 10 and Figure 11). The ODSA web server can be used to present different subscription options to the end-user and capture Terms & Conditions agreements.

The product implementations for the Entitlement Configuration Server and the Service Provider's portal web server shall protect the privacy of the subscriber and of the end-user on all data that could be used for tracking such as ICCID, MSISDN, EID.

Instead of just one entitlement configuration request, the ODSA application requires several exchanges with the Entitlement Configuration Server. Each exchange is associated with an operation, resulting in the need of a new string-based `operation` request parameter.

Table 26 presents the allowed operations for the eSIM ODSA procedure.

ODSA Operation	Description
`CheckEligibility`	To verify if end-user is allowed to invoke the ODSA application
`ManageSubscription`	To request for subscription-related action on a primary or companion device.
`ManageService`	To activate / deactivate the service on the primary or companion device. This is an optional operation.
`AcquireConfiguration`	To provide service-related data about a primary or companion device.

ODSA Operation	Description
`AcquirePlan`	To request available plans to be offered by the MNO to a specific user or MDM
`AcquireTemporaryToken`	To request a Temporary Token from the ECS, to allow authentication for a device that may not have the means to acquire the TOKEN

Table 26. ODSA Operations

6.2 ODSA Request Parameters

The ODSA procedure for Primary and Companion devices requires additional parameters in the HTTP requests, outside of the ones described in 2.2. Table 27 presents the new parameters and their associated ODSA operations.

New GET parameters for ODSA application	Type	Values	Description
operation	String	`CheckEligibility` , `ManageSubscription`, `ManageService`, `AcquireConfiguration`, `AcquirePlan`, `AcquireTemporaryToken`, `GetPhoneNumber`, `VerifyPhoneNumber`, `GetSubscriberInfo`	Indicates the operation requested by the “ODSA for eSIM device” application
operation_type	Integer	Used by the `ManageSubscription` operation.	
		0 - SUBSCRIBE	to activate a subscription for the eSIM device.
		1 - UNSUBSCRIBE	to cancel a subscription for the eSIM device.
		2 – CHANGE SUBSCRIPTION	to manage an existing subscription on the eSIM device.
		3 – TRANSFER SUBSCRIPTION	to transfer a subscription from an existing device (with physical SIM or eSIM) to the eSIM device
		4 – UPDATE SUBSCRIPTION	to inform the network of a subscription update on the eSIM device
		5 – ACTIVATE TERMINAL ICCID	to inform the network that the terminal_iccid or companion_terminal_iccid which is in a ServiceStatus DEACTIVATED state can be moved to an ACTIVATED state
		6 – DEACTIVATE TERMINAL ICCID	to inform the network that the terminal_iccid or companion_terminal_iccid which is in a ServiceStatus ACTIVATED state can be moved to a DEACTIVATED state

New GET parameters for ODSA application	Type	Values	Description
		7 – ACTIVE SUBSCRIPTION RECOVER	to inform the network that the eSIM profile represented by the "terminal_iccid" has been removed by the end user via factory reset or other operations, the end user would like to recover the active subscription associated to it.
		Used by the `ManageService` operation.	
		10 – ACTIVATE SERVICE	Indicates this is a request to activate a service on the eSIM device.
		11 – DEACTIVATE SERVICE	Indicates this is a request to deactivate a service on the eSIM device.
operation_targets	String	Used by the `AcquireTemporaryToken` operation.	
		Comma separated list of the operation field found in this table	To acquire a temporary token associated with the ODSA operation(s) and AppID.
companion_terminal_id	String	Used by **all** the Companion ODSA operations.	
		Any string value	This value shall be a unique and persistent identifier of the device. This identifier may be an IMEI (preferred) or a UUID.
companion_terminal_vendor (Conditional)	String	Used by the operations `CheckEligibility`, `ManageSubscription` and `ManageService` for Companion ODSA. It shall be present in a `ManageSubscription` request.	
		Any string value	Manufacturer of the companion device.
companion_terminal_model (Optional)	String	Used by the operations `CheckEligibility`, `ManageSubscription` and `ManageService` for Companion ODSA.	
		Any string value	Model of the companion device.
companion_terminal_sw_version (Optional)	String	Used by the operations `CheckEligibility`, `ManageSubscription` and `ManageService` for Companion ODSA.	
		Any string value	Software version of the companion device.
companion_terminal_friendly_name (Conditional)	String	Used by the operations `CheckEligibility`, `ManageSubscription` and `ManageService` for Companion ODSA. It shall be present in a `ManageSubscription` request during the device activation flow.	
		Any string value	User-friendly identification for the companion device which can be used by the Service Provider in Web Views.
companion_terminal_service	String	Used by the `ManageSubscription` and `ManageService` operation for Companion ODSA.	

New GET parameters for ODSA application	Type	Values	Description
(Conditional)		`SharedNumber`	Indicates that the service being managed is “Shared Number”, where the companion device carries the same MSISDN as the primary device. This parameter shall be included as part of the ManageService operation in order to indicate which service is being managed. It is optional to include as part of the ManageSubscription operation.
		`DiffNumber`	Indicates that the service being managed is “Different Number”, where the companion device carries a different MSISDN from the primary device but is assigned to the same subscriber. This parameter shall be included as part of the ManageService operation in order to indicate which service is being managed. It is optional to include as part of the ManageSubscription operation.
		`FamilyNumber`	Indicates that the service being managed is “Family Number”, where the companion device carries a different MSISDN from the primary device and the MSISDN can be assigned to another individual or subscriber. This parameter shall be included as part of the ManageService operation in order to indicate which service is being managed. It is optional to include as part of the ManageSubscription operation.
companion_terminal_iccid (Conditional)	String	Used by the `ManageSubscription, ManageService` and `AcquireConfiguration` operations for Companion ODSA.	
		Value following the ICCID format	The ICCID of the companion device being managed, provided only if there is an eSIM profile on the companion’s eUICC. This parameter shall be included in the ManageService operation to indicate which ICCID is being managed. It is optional to include this parameter as part of the ManageSubscription and AcquireConfiguration operations.
companion_terminal_eid (Conditional)	String	Used by the `ManageSubscription` and `AcquireConfiguration` operations for Companion ODSA. It shall be present in a `ManageSubscription` request.	
		Value following eUICC format	eUICC identifier (EID) of the companion device being managed

New GET parameters for ODSA application	Type	Values	Description
old_companion_terminal_id (Conditional)	String	Used by the `ManageSubscription` operation for Companion ODSA when the user selected an `old_companion_terminal_id` using a Companion ODSA client that's supports a standalone eSIM management MMI.	
		Any string value	A unique identifier for the companion device. Suggested source is the IMEI of the device.
old_companion_terminal_iccid (Conditional)	String	Used by the `ManageSubscription` operation for Companion ODSA when the user selected an `old_companion_terminal_iccid` using a Companion ODSA client that's supports a standalone eSIM management MMI.	
		Any string value	The old ICCID of the companion device being managed, provided only if there is an eSIM profile on the companion's eUICC
terminal_iccid (Optional)	String	Used by the `ManageSubscription` and `AcquireConfiguration` operations for Primary ODSA, in case a primary SIM is not accessible (or not present). `terminal_id` is associated with the device or eSIM being managed.	
		Any string value	The ICCID of the primary eSIM being managed
terminal_eid (Optional)	String	Used by the `ManageSubscription` and `AcquireConfiguration` operations for Primary ODSA, in case a primary SIM is not accessible (or not present). `terminal_id` is associated with the device or eSIM being managed.	
		Value following eUICC format	eUICC identifier (EID) of the primary eSIM being managed
target_terminal_entitlement_protocol (Conditional)	Integer	Used by the `CheckEligibility and AcquireTemporaryToken`, operations for Primary ODSA when subscription transfer is for cross-TS.43 platform	
		`0 - ts43`	Device supports TS.43 entitlement protocol
		`1 - other`	Device doesn't support TS.43 entitlement protocol
target_terminal_id (Conditional)	String	Used by the `CheckEligibility`, `ManageSubscription` and `AcquireConfiguration` operations for Primary ODSA. This parameter provides the identity of the eSIM being managed. For the transfer subscription use case, this parameter (ID) is expected to be the **IMEI** of the new/targeted device.	
		Any string value	This value shall be a unique and persistent identifier of the eUICC being managed. This identifier may be an IMEI associated with the eUICC.
target_terminal_iccid (Optional)	String	Used by the `ManageSubscription` and `AcquireConfiguration` operations for Primary ODSA	

New GET parameters for ODSA application	Type	Values	Description
		Value following the ICCID format	The ICCID of the primary eSIM being managed
old_terminal_entitlement_protocol (Conditional)	Integer	Used by `ManageSubscription` and `AcquireConfiguration` operations for Primary ODSA when subscription transfer is for cross-TS.43 platform	
		`0 - ts43`	Device supports TS.43 entitlement protocol
		`1 - other`	Device doesn't support TS.43 entitlement protocol
target_terminal_eid (Optional)	String	Used by the `ManageSubscription` and `AcquireConfiguration` operations for Primary ODSA. For the transfer subscription use case, this parameter (EID) is expected to be the **EID** of the new/target device.	
		Value following eUICC format	eUICC identifier (EID) of the primary eSIM being managed
old_terminal_id (Optional)	String	Used by the `ManageSubscription/Transfer Subscription` for Primary ODSA in case the request is created by an old primary device.	
		Value following `terminal_id` format	The unique identifier, for example IMEI (preferred) or a UUID for the old primary device.
old_terminal_iccid (Optional)	String	Used by the `ManageSubscription/Transfer Subscription` for Primary ODSA in case the request is created by an old primary device.	
		Value following the ICCID format	The Profile's ICCID of an old primary device to be selected by an end-user for subscription transfer to a new primary device.
redownloadable_profile (Optional)	Integer	Used by the `ManageSubscription/Transfer Subscription` for Primary ODSA to identify if the device supports eSIM Transfer with redownloadable profile.	
		0 – NOT SUPPORTED	Device doesn't support redownloadable profile flow
		1 – SUPPORTED	Device supports redownloadable profile flow
enterprise_id (Optional)	String	Used by the operations `CheckEligibility` for server-initiated ODSA	
		Any string value	Identifier provided by the MNO to identify the enterprise
enterprise_terminal_id (Optional)	String	Used by the `ManageSubscription` and `AcquireConfiguration` operations for server-initiated ODSA.	
		Any string value	This value shall be a unique and persistent identifier of the enterprise device. This identifier may be an IMEI (preferred).
enterprise_terminal_eid	String	Used by the `ManageSubscription` and `AcquireConfiguration` operations for server-initiated ODSA.	

New GET parameters for ODSA application	Type	Values	Description
(Optional)		Any string value	eUICC identifier (EID) of the device being managed
plan_id (Optional)	String	Used by the operations `ManageSubscription` for server-initiated ODSA to identify the selected plan for a specific subscriber identified by `enterprise_terminal_id` and `enterprise_terminal_eid`	
		Any string value	Identifier of the specific plan offered by an MNO
MSG_btn (Conditional)	Integer	Used by the `ManageSubscription` operation for Primary ODSA. This indicates either “Accept” or “Reject” button has been pressed on the device UI.	
		0 – REJECTED	MSG content has been rejected by the user.
		1 – ACCEPTED	MSG content has been accepted by the user.
MSG_response (Conditional)	String	Used by the `ManageSubscription` operation for Primary ODSA. This indicates the response entered by the user on the device UI. This field shall only be present if user ACCEPTED, and the user has entered a value.	
		Any string value	Value entered by the user.
MSG_character_display_limits (Optional)	List of Integers	Used by the `ManageSubscription` and `AcquireConfiguration` during an ODSA operation. A comma-separated ordered list of non-zero, positive integers representing of the character limits the client application can display to the user without modification. If there is no limit, the value of -1 shall be sent.	
		-1 or non-Zero Integer value	Title character limit is in the 1st position of the list.
		-1 or non-Zero Integer value	Message character limit is in the 2nd position of the list.
		-1 or non-Zero Integer value	Accept_btn_label character limit is in the 3rd position of the list.
		-1 or non-Zero Integer value	Reject_btn_label character limit is in the 4th position of the list.
		-1 or non-Zero Integer value	Accept_freetext_hint character limit is in the 5th position of the list.
		-1 or non-Zero Integer value	Accept_freetext_validation_failed_error_text character limit is in the 6th position of the list.
msisdn (Conditional)	String	Used by the `VerifyPhoneNumber` operation to compare this value with the one mapped to the token generated during the Authentication process.	
		MSISDN of the subscription in E.164 format.	MSISDN to verify.

Table 27. New parameters for ODSA application

6.3 Devices Identifiers used for Request Parameters

Table 4 and Table 27 present a number of identity parameters (ending with `_ID`, `_id`, `_eid` or `_iccid`) that need to be associated with an identifier on the primary or companion

device. The following offers the mapping between device identifiers and identity parameters for companion and primary ODSA use cases and their different operations.

Figure 12 shows the suggested association between identity parameters ←→ device identifiers for the Companion ODSA use case where a requesting device's SIM is accessible. Authentication is performed using EAP-AKA with that SIM.

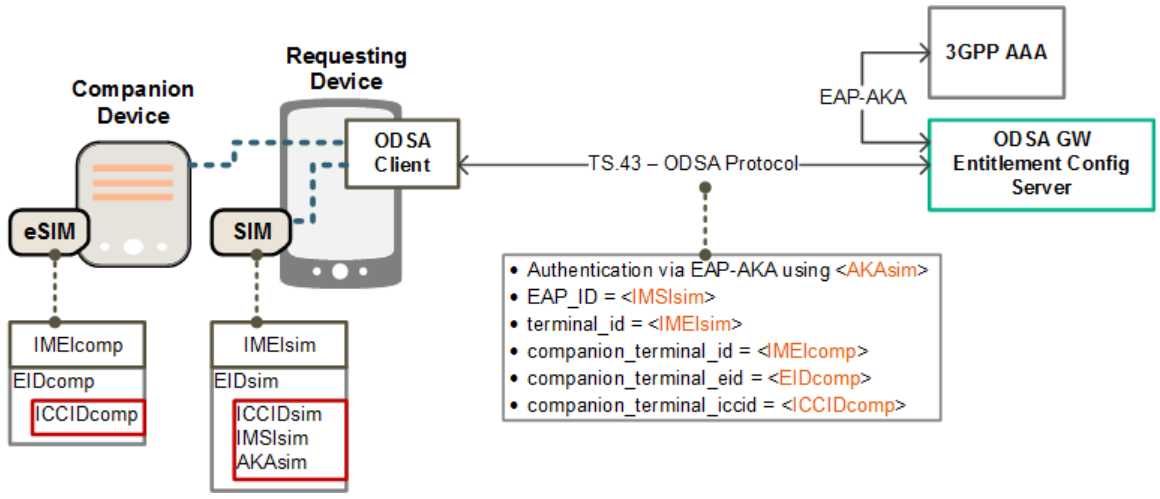

Figure 12. Identifier Mapping for Companion ODSA with access to SIM on Requesting Device

Figure 13 shows the suggested association between identity parameters ←→ device identifiers for the Companion ODSA use case where a SIM on the requesting device is not accessible. Authentication is performed using OAuth 2.0 / OIDC. Note the use of the application's UUID in case the requesting device's IMEI is not known.

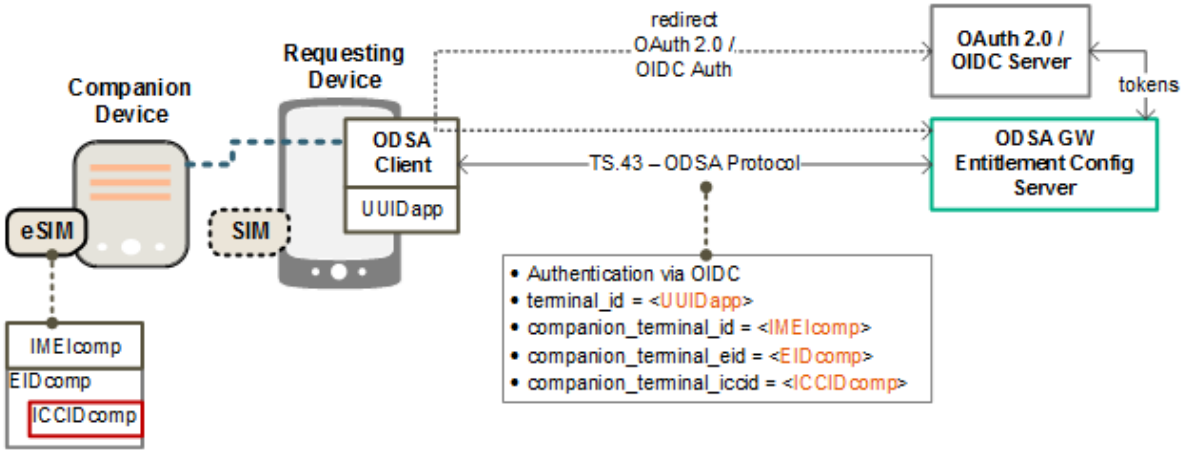

Figure 13. Identifier Mapping for Companion ODSA when Requesting Device's SIM is not present or accessible.

Figure 14 shows the suggested association between identity parameters ←→ device identifiers for the Primary ODSA use case where a SIM on the device that belongs to the Service Provider is accessible. Authentication is performed using EAP-AKA with that SIM.

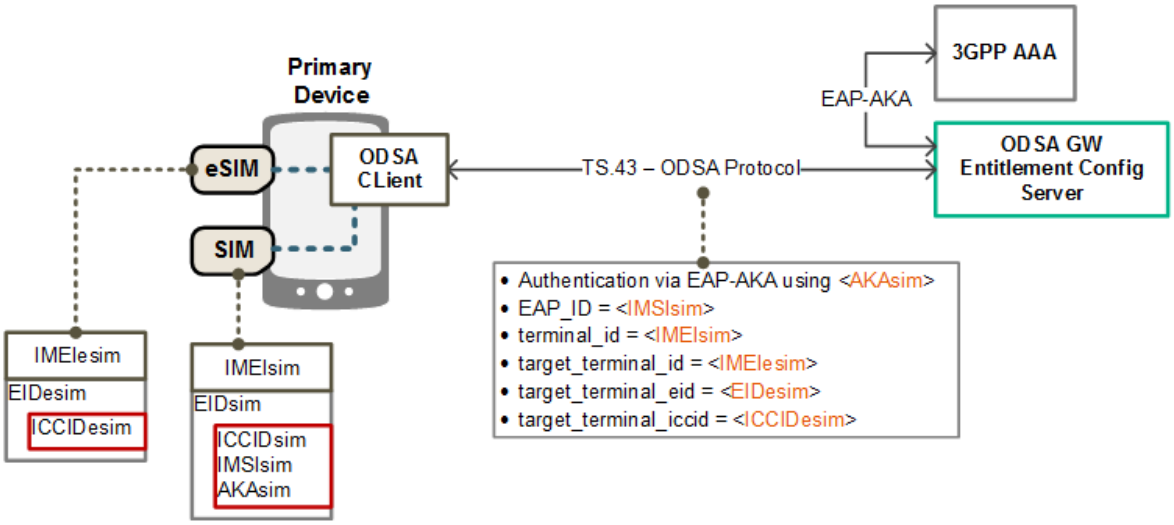

Figure 14. Identifier Mapping for Primary ODSA with access to a SIM

Figure 15 shows the suggested association between identity parameters ←→ device identifiers for the Primary ODSA use case where the data and AKA of a primary SIM is not accessible (or not present). Authentication is performed using OAuth 2.0 / OIDC.

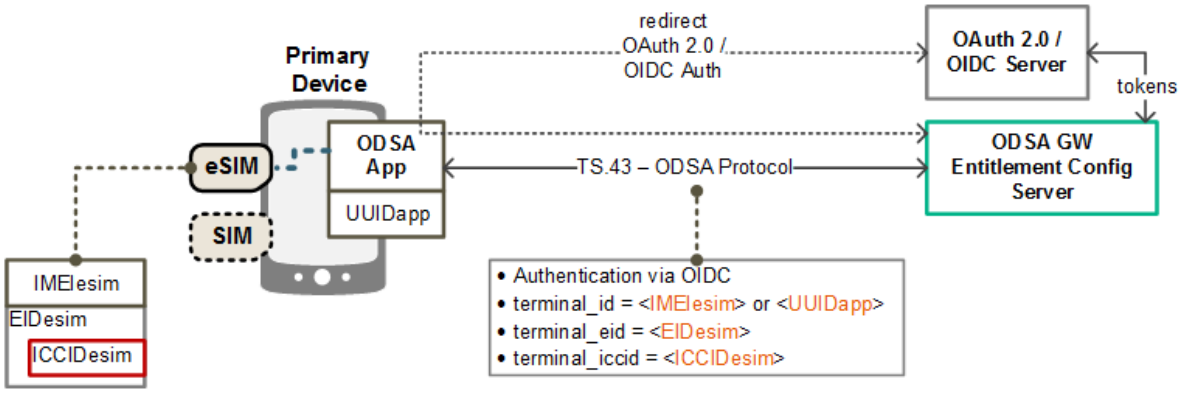

Figure 15. Identifier Mapping for Primary ODSA when existing SIM data is not accessible or not present.

Figure 16 shows the suggested association between identity parameters ←→ device/server identifiers for the Server-Initiated ODSA use case. Authentication is performed using server-to-server OAuth 2.0 as described in section 2.8.3.

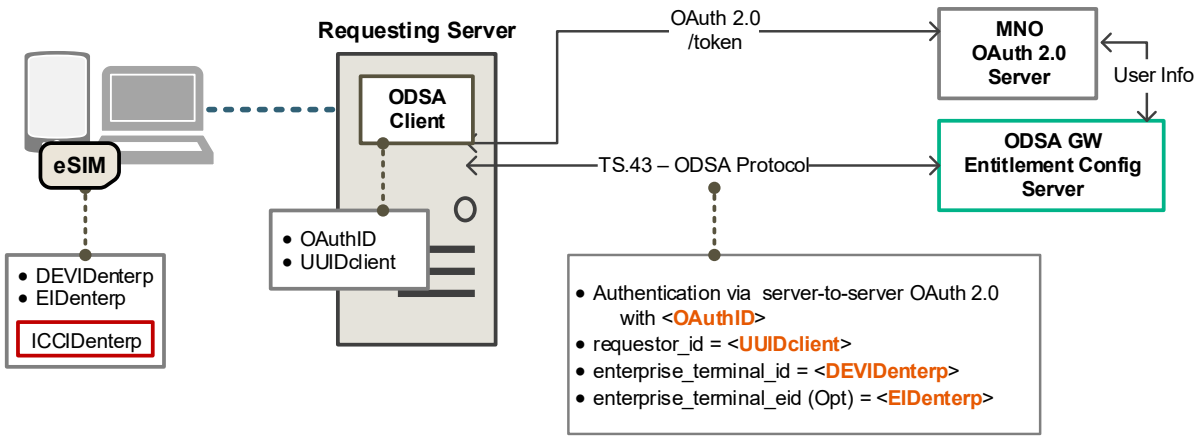

Figure 16. Identifier Mapping for Server-initiated ODSA

Figure 17 shows the suggested association between identity parameters ←→ device identifiers for the Primary ODSA use case with Subscription Transfer where the old Primary Device's SIM data is accessible. Authentication is performed using EAP-AKA with that SIM.

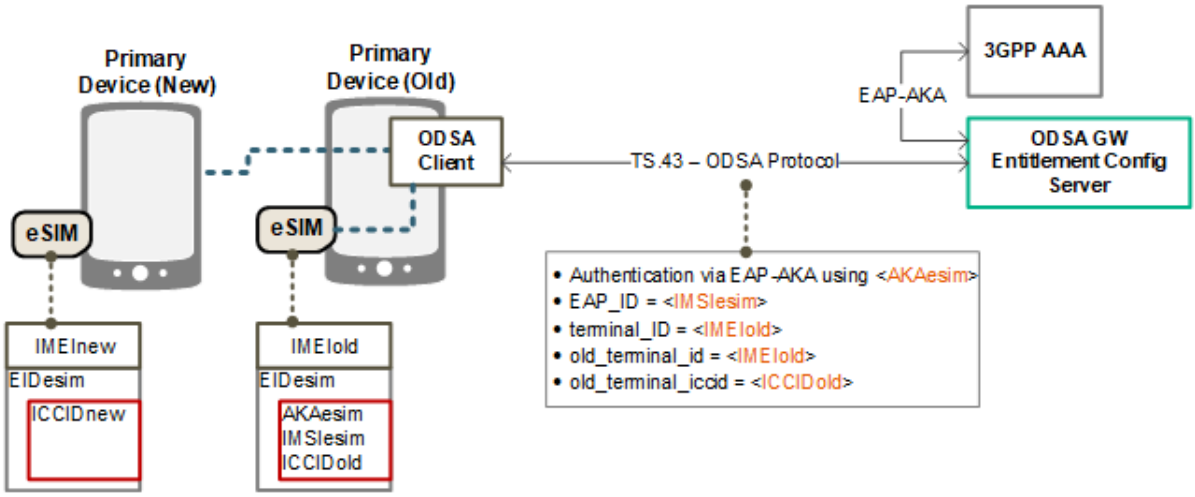

Figure 17: Identifier Mapping for Primary ODSA when Requesting Device's SIM data is accessible.

6.4 Examples of ODSA Requests

This section presents samples of ODSA requests using the GET method. It is also possible to use the POST method as indicated in section 2.4. In the POST case, the parameters would be located in the message body as a JSON object instead of being in the HTTP query string.

6.4.1 CheckEligibility Request Example

Table 28 presents an example for the CheckEligibility operation for an ODSA application.

```
GET ? terminal_id = 013787006099944&
token = es7w1erXjh%2FEC%2FP8BV44SBmVipg&
terminal_vendor = TVENDOR&
terminal_model = TMODEL&
terminal_sw_version = TSWVERS&
entitlement_version = ENTVERS&
app = ap2006&
operation = CheckEligibility&
companion_terminal_id = 98112687006099944&
vers = 1 HTTP/1.1

Host: entitlement.telco.net:9014
User-Agent: PRD-TS43 TVENDOR/TMODEL Primary-ODSA/TSWVERS OS-Android/8.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Table 28. Example of a CheckEligibility ODSA Request

6.4.2 ManageSubscription Request Example

Table 29 presents an example for the Manage Subscription operation for an ODSA application.

```
GET ? terminal_id = 013787006099944&
token = es7w1erXjh%2FEC%2FP8BV44SBmVipg&
entitlement_version = ENTVERS
app = ap2006&
operation = ManageSubscription&
operation_type = 0&                                  ! subscribe
companion_terminal_id = 98112687006099944&
companion_terminal_eid = JHSDHljhsdfy763hh&
vers = 1 HTTP/1.1

Host: entitlement.telco.net:9014
User-Agent: PRD-TS43 TVENDOR/TMODEL Primary-ODSA/TSWVERS OS-Android/8.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Table 29. Example of a ManageSubscription ODSA Request

6.4.3 ManageService Request Example

Table 30 presents an example for the Manage Service operation for an ODSA application.

```
GET ? terminal_id = 013787006099944&
token = es7w1erXjh%2FEC%2FP8BV44SBmVipg&
terminal_vendor = TVENDOR&
terminal_model = TMODEL&
terminal_sw_version = TSWVERS&
entitlement_version = ENTVERS&
app = ap2006&
operation = ManageService&
operation_type = 10&                                ! activate service
companion_terminal_id = 98112687006099944&
companion_terminal_service = DiffNumber&
companion_terminal_iccid = 89000123766789001878&
vers = 1 HTTP/1.1

Host: entitlement.telco.net:9014
User-Agent: PRD-TS43 TVENDOR/TMODEL Primary-ODSA/TSWVERS OS-Android/8.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Table 30. Example of a ManageService ODSA Request

6.4.4 AcquireConfiguration Request Example

Table 31 presents an example for the Acquire Configuration operation for an ODSA application.

```
GET ? terminal_id = 013787006099944&
token = es7w1erXjh%2FEC%2FP8BV44SBmVipg&
terminal_vendor = TVENDOR&
terminal_model = TMODEL&
terminal_sw_version = TSWVERS&
entitlement_version = ENTVERS&
app = ap2006&
operation = AcquireConfiguration&
companion_terminal_id = 98112687006099944&
vers = 1 HTTP/1.1
MSG_character_display_limits=55,270,20,20,40,45

Host: entitlement.telco.net:9014
User-Agent: PRD-TS43 TVENDOR/TMODEL Primary-ODSA/TSWVERS OS-Android/8.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Table 31. Example of an AcquireConfiguration ODSA Request

6.4.5 AcquirePlan Request Example

Table 32 presents an example for the AcquirePlan operation for a server ODSA application.

```
GET ? requestor_id = 06170799658&
token = es7w1erXjh%2FEC%2FP8BV44SBmVipg&
terminal_vendor = TVENDOR&
terminal_model = TMODEL&
terminal_sw_version = TSWVERS&
entitlement_version = ENTVERS&
app = ap2011&
operation = AcquirePlan&                          ! get plans
vers = 1 HTTP/1.1

Host: entitlement.telco.net:9014
User-Agent: PRD-TS43 TVENDOR/TMODEL Primary-ODSA/TSWVERS OS-Android/8.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Table 32. Example of an AcquirePlan ODSA Request

6.4.6 AcquireTemporaryToken Request Example

Table 33 presents an example for the `AcquireTemporaryToken` operation for a server ODSA application.

```
GET ? terminal_id = 06170799658&
token = es7w1erXjh%2FEC%2FP8BV44SBmVipg&
terminal_vendor = TVENDOR&
terminal_model = TMODEL&
terminal_sw_version = TSWVERS&
entitlement_version = ENTVERS&
terminal_iccid = 98151515135132135135 13&
operation_targets = ManageSubscription%2CAcquireConfiguration&
app = ap2009&
operation = AcquireTemporaryToken&
vers = 1 HTTP/1.1

Host: entitlement.telco.net:9014
User-Agent: PRD-TS43 TVENDOR/TMODEL Primary-ODSA/TSWVERS OS-Android/8.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Table 33. Example of an AcquireTemporaryToken ODSA Request

6.4.7 GetPhoneNumber Request Example

Following sections provides some examples depending on the device sending the `getPhoneNumber` request (device or application server).

6.4.7.1 GetPhoneNumber request for client

Table 34 presents an example for the GetPhoneNumber operation for a primary client.

```
GET ? terminal_id = 09999799658&
token = es7w1erXjh%2FEC%2FP8BV44SBmVipg &
terminal_vendor = TVENDOR&
terminal_model = TMODEL&
terminal_sw_version = TSWVERS&
entitlement_version = ENTVERS&
app = ap2014&
operation = GetPhoneNumber&
vers = 1 HTTP/1.1

Host: entitlement.telco.net:9014
User-Agent: PRD-TS43 TVENDOR/TMODEL Primary-ODSA/TSWVERS OS-Android/8.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Table 34. Example of an GetPhoneNumber primary client Request

6.4.7.2 GetPhoneNumber request sent by application server.

Table 35 presents an example for the GetPhoneNumber operation for an application server.

```
GET ? requestor_id = 06170799658&
temporary_token = es7w1erXjh%2FEC%2FP8BV44SBmVipg&
access_token = 32487234987238974& //OPTIONAL
terminal_vendor = TVENDOR&
terminal_model = TMODEL&
terminal_sw_version = TSWVERS&
entitlement_version = ENTVERS&
app = ap2014&
operation = GetPhoneNumber&
vers = 1 HTTP/1.1

Host: entitlement.telco.net:9014
User-Agent: PRD-TS43 TVENDOR/TMODEL Primary-ODSA/TSWVERS OS-Android/8.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Table 35. Example of an GetPhoneNumber application server Request

6.4.8 VerifyPhoneNumber Request Example

Table 36 presents an example for the `VerifyPhoneNumber` operation.

```
GET ? terminal_id = 09999799658&
token = es7w1erXjh%2FEC%2FP8BV44SBmVipg &
terminal_vendor = TVENDOR&
terminal_model = TMODEL&
terminal_sw_version = TSWVERS&
entitlement_version = ENTVERS&
app = ap2014&
operation = VerifyPhoneNumber&
msisdn = <MSISDN>&
vers = 1 HTTP/1.1

Host: entitlement.telco.net:9014
User-Agent: PRD-TS43 TVENDOR/TMODEL Primary-ODSA/TSWVERS OS-Android/8.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Table 36. Example of an VerifyPhoneNumber Request

6.4.9 GetSubscriberInfo Request Example

Table 37 presents an example for the `GetSubscriberInfo` operation for an application server.

```
GET ? requestor_id = 06170799658&
temporary_token = es7w1erXjh%2FEC%2FP8BV44SBmVipg&
access_token = 32487234987238974& //OPTIONAL
terminal_vendor = TVENDOR&
terminal_model = TMODEL&
terminal_sw_version = TSWVERS&
entitlement_version = ENTVERS&
app = ap2014&
operation = GetSubscriberInfo&
vers = 1 HTTP/1.1

Host: entitlement.telco.net:9014
User-Agent: PRD-TS43 TVENDOR/TMODEL Primary-ODSA/TSWVERS OS-Android/8.0 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Table 37. Example of a GetSubscriberInfo Application Server Request

6.5 ODSA Configuration Parameters

6.5.1 General / Always-Present Configuration Parameters

- Parameter names:
    - `OperationResult`: Mandatory
    - `GeneralErrorURL`: Optional
    - `GeneralErrorUserData`: Optional
    - `GeneralErrorText`: Optional

The `OperationResult` parameter provides the result of the requested operation as described in Table 38.

The URL and User Data parameters offer the option of using operator-specific web views when the end-user OIDC authentication process fails. If `GeneralErrorText` is present (and the URL and User Data are missing) the device presents the text to the end-user. If all fields are absent, the device presents instead an internally-generated message to the end-user.

General Configuration Parameter	Type	Values	Description
OperationResult (Mandatory)	Integer	1 - SUCCESS	Operation was a success
		100 - ERROR, GENERAL	There was a general error during processing. Device shall stop the execution of current ODSA procedure.
		101 - ERROR, INVALID OPERATION	An invalid operation value was provided in request. Device shall stop executing ODSA procedure.
		102 - ERROR, INVALID PARAMETER	An invalid parameter name or value was provided in request. Device shall stop executing ODSA procedure.
		103 - WARNING, NOT SUPPORTED OPERATION	The optional operation is not supported by the carrier. Device should continue with the flow. This error only applies to optional operations (for example ManageService).
		104 – ERROR, INVALID MSG RESPONSE	The contents of the `MSG_response` are incorrect or unexpected.
GeneralErrorURL (Optional)	String	URL to a Service Provider site or portal	The provided URL shall present a Web view to user on the reason(s) why the authentication failed.
GeneralErrorUserData (Optional)	String	Parameters or content to insert when invoking URL provided in the `GeneralErrorURL` parameter	User data sent to the Service Provider when requesting the `GeneralErrorURL` web view. It should contain user-specific attributes to improve user experience.
GeneralErrorText (Optional)	String	Any string value	User-specific content string to be shown to the user.

Table 38. General Configuration Parameters for ODSA Operation

6.5.2 CheckEligibility Operation Configuration Parameters

- Parameter names and presence:
  - `CompanionAppEligibility`: Mandatory for Companion ODSA
  - `PrimaryAppEligibility`: Mandatory for Primary ODSA
  - `EnterpriseAppEligibility`: Mandatory for server-initiated ODSA
  - `CompanionDeviceServices`: Mandatory for Companion ODSA
  - `NotEnabledURL`: Optional

- `NotEnabledUserData`: Optional
- `NotEnabledContentsType`: Optional

Those parameters are associated with the eligibility of offering the ODSA application on the requesting device and for the end-user. The application usually runs on the primary device (with SIM or eSIM). The eligibility value can be based on factors like the type of end-user's subscription/plans and the device details.

The `CompanionDeviceServices` parameter represents the different services that can be activated on the companion device.

The URL, User Data and Contents Type parameters offer the option of using operator-specific web views when the end-user attempts to invoke the Companion or Primary ODSA application when it is not enabled. If absent, the device presents instead an internally-generated message to the end-user.

The different values for the configuration parameters of the `CheckEligibility` operation are provided in Table 39.

"Check Eligibility" Configuration parameter	Type	Values	Description
CompanionAppEligibility or PrimaryAppEligibility or EnterpriseAppEligbility	Integer	0 - DISABLED	ODSA app cannot be offered and invoked by the end-user or server (for a specific enterprise_id)
		1 - ENABLED	ODSA app can be invoked by end-user or server (for a specific enterprise_id) to activate a new subscription
		2 - INCOMPATIBLE	ODSA app is not compatible with the device or server
CompanionDeviceServices (Mandatory)	String	Comma-separated list with all services available on the companion device	
		`SharedNumber`	Indicates that the "Shared Number" service is active on the companion device (where the device carries the same MSISDN as the primary one)
		`DiffNumber`	Indicates that the "Diff Number" service is active on the companion device (where the device carries a different MSISDN from the primary one but is assigned to the same subscriber.)
		`FamilyNumber`	Indicates that the configuration is for the "Family Number" service (where the device carries a different MSISDN from the primary one and the MSISDN can be assigned to another individual or subscriber.)
NotEnabledURL (Optional)	String	URL to a Service Provider site or portal	The provided URL shall present a Web view to user on the reason(s) why the ODSA app cannot be used/invoked

"Check Eligibility" Configuration parameter	Type	Values	Description
NotEnabledUserData (Optional)	String	Parameters or content to insert when invoking URL provided in the `NotEnabledURL` parameter	User data sent to the Service Provider when requesting the `NotEnabledURL` web view. It should contain user-specific attributes to improve user experience. The format must follow the `NotEnabledContentsType` parameter. For content types of JSON and XML, it is possible to provide the base64 encoding of the value by preceding it with `encodedValue=`.
NotEnabledContentsType (Optional)	String	Specifies content and HTTP method to use when reaching out to the web server specified in `NotEnabledURL`.	
		NOT present	Method to `NotEnabledURL` is HTTP GET request with query parameters from `NotEnabledUserData`.
		`json`	Method to `NotEnabledURL` is HTTP POST request with JSON content from `NotEnabledUserData`.
		`xml`	Method to `NotEnabledURL` is HTTP POST request with XML content from `NotEnabledUserData`.
PollingInterval (Optional)	Integer	A valid positive integer number including 0 value.	Specifies the minimum interval with which the client application may poll the ECS to refresh the current `PrimaryAppEligiblity` using the `CheckEligibility` request. This parameter may be present only when `PrimaryAppEligibility`=0 – DISABLED. If parameter is not present or value=0, this polling procedure is not triggered and ODSA App will keep waiting for any external action to continue the flow. The maximum number of `CheckEligibilty` requests will be defined as an ECS configuration variable (`MaxRefreshRequest`)
PollingIntervalUnit (Optional)	Integer	0 – minutes 1 – seconds 2 – deciseconds	Specifies the time unit for the PollingInterval parameter. If this parameter is not present, 0 – minutes will be considered as default value

Table 39. Configuration Parameters – Check Eligibility ODSA Operation

6.5.3 ManageSubscription Operation Configuration Parameters

- Parameter names and presence:
  - `SubscriptionResult`: Mandatory

- `SubscriptionServiceURL`: Conditional
- `SubscriptionServiceUserData`: Conditional
- `SubscriptionServiceContentsType`: Conditional
- `DownloadInfo`: Conditional
- `MSG`: Optional for Primary ODSA

Those parameters provide the result of an ODSA subscription request, including any additional data needed to complete the subscription (URL to send users to, or eSIM profile download information for the eSIM device).

The ECS may include an `MSG` structure in order to communicate terms and conditions to the user or query information from the user without a webview.

The different values for the configuration parameters of the `ManageSubscription` operation are provided in Table 40.

"ManageSubscription" Configuration parameters	**Type**	**Values**	**Description**
SubscriptionResult (Mandatory)	Integer	1 - CONTINUE TO WEBSHEET	Indicates that end-user must go through the subscription web view procedure, using information included below.
		2 - DOWNLOAD PROFILE	Indicates that an eSIM profile must be downloaded by the device, with further information included in response
		3 – DONE	Indicates that subscription flow has ended, and the end-user has already downloaded the eSIM profile so there is no need to perform any other action.
		4 - DELAYED DOWNLOAD	Indicates that an eSIM profile is not ready to be downloaded when a user requests to transfer subscription or to add the new subscription through native UX on the eSIM device.
		5 – DISMISS	Indicates that subscription flow has ended without completing the ODSA procedure. An eSIM profile is not available.
		6 - DELETE PROFILE IN USE	Indicates that the profile in use needs to be deleted to complete the subscription transfer.
		7 – REDOWNLOADABLE PROFILE IS MANDATORY	Indicates that implementing redownloadable profile is mandatory. If device is not able to support this, it should end the process. This parameter only applies when operation_type=3 (transfer subscription)
		8 – REQUIRES USER INPUT	Indicates that user input without a webview is required in order to complete the operation_type requested with the information submitted to the ECS.

"ManageSubscription" Configuration parameters	Type	Values	Description
SubscriptionServiceURL (Conditional)	String	URL to a Service Provider site or portal	Present only if `SubscriptionResult` is "1". URL refers to web views responsible for a certain action on the eSIM device subscription. The Service Provider can provide different URL based on the `operation_type` input parameter (subscribe, unsubscribe, change subscription).
SubscriptionServiceUserData (Conditional)	String	Parameters to insert when invoking URL provided in `SubscriptionServiceURL`	Present only if `SubscriptionResult` is "1", and also optional. User data sent to the Service Provider when requesting the `SubscriptionServiceURL` web view. It should contain user-specific attributes to improve user experience. The format must follow `SubscriptionServiceContentsType`. For content types of JSON and XML, it is possible to provide the base64 encoding of the value by preceding it with `encodedValue=`.
SubscriptionService ContentsType (Conditional)	String	Specifies content and HTTP method to use when reaching out to the web server specified by `SubscriptionServiceURL`	
		NOT present	Method to `SubscriptionServiceURL` is HTTP GET request with query parameters from `SubscriptionServiceUserData`.
		"`json`"	Method to `SubscriptionServiceURL` is HTTP POST request with JSON content from `SubscriptionServiceUserData`.
		"`xml`"	Method to `SubscriptionServiceURL` is HTTP POST request with XML content from `SubscriptionServiceUserData`.
DownloadInfo (Conditional)	Structure	multi-parameter value - see next table for details	Present if `SubscriptionResult` is "2". Specifies how and where to download the eSIM profile associated with the companion or primary device.
MSG (Optional)	Structure	multi-parameter value – see Table 45 for details	Includes information to be communicated and displayed to the user in order to complete the operation_type currently being requested by the client application. Only present if SubscriptionResult is set to 8 – REQUIRES USER INPUT

Table 40. Configuration Parameters – Manage Subscription ODSA Operation

The `DownloadInfo` configuration parameter is defined as a structure with several parameters as shown in Table 41.

"Download Info" parameters	Type	Values	Description
ProfileIccid (Conditional)	String	ICCID of the eSIM profile to download from SM-DP+	The ICCID shall be included in the case where ProfileSmdpAddress is used to trigger the profile download. Can be a new ICCID or the re-usable ICCID that was provided in the request parameter `companion_terminal_iccid` or `target_terminal_iccid`
ProfileSmdpAddress (Conditional)	String	FQDN to SM-DP+ platform of MNO	Address(es) of SM-DP+ to obtain eSIM profile. If more than one, they must be comma-separated. It is not needed if `ProfileActivationCode` is present. **Note**: for this download method to be used, the client must provide the EID of the eSIM in the request, as either `terminal_eid`, `companion_terminal_eid` or `target_terminal_eid` as defined in Table 27.
ProfileActivationCode (Conditional)	String	Encoded in Base64. Must follow the activation code format from GSMA SGP.22	Activation code as defined in SGP.22 to permit the download of an eSIM profile from an SM-DP+. It is not needed if `ProfileSmdpAddress` is present.

Table 41. Configuration Parameters – Download Info for Manage Subscription

6.5.4 ManageService Operation Configuration Parameters

- Parameter names and presence:
  - `ServiceStatus`: Mandatory

The parameter provides the result of an ODSA service request.

The different values for the configuration parameters of the `ManageService` operation are provided in Table 42.

"ManageService" Configuration parameters	Type	Values	Description
ServiceStatus (M)	Integer	1 - ACTIVATED	eSIM device's service is activated.
		2 - ACTIVATING	eSIM device's service is being activated.
		3 - DEACTIVATED	eSIM device's service is not activated.
		4 - DEACTIVATED, NO REUSE	eSIM device's service is not activated and the associated ICCID should not be reused.

Table 42. Configuration Parameters – Manage Service ODSA Operation

6.5.5 AcquireConfiguration Operation Configuration Parameters

- Parameter names and presence:
    - `CompanionConfigurations`: Conditional for Companion ODSA, Top level, present if there is one or more companion device(s) associated with the requesting device that carry a configuration for ODSA.
    - `CompanionConfiguration`: Within `CompanionConfigurations`, one or more
    - `PrimaryConfigurations`: Conditional for Primary ODSA, Top level, present if one or more `PrimaryConfiguration`(s) managed by the ECS are associated with the requesting subscription. Where one `PrimaryConfiguration` is designated the Primary ICCID or eSIM profile by the ECS. The remaining Configurations are to be designated as the Secondary ICCID(s) or eSIM profile(s).
    - `PrimaryConfiguration`: Mandatory for Primary ODSA. This parameter can be used stand-alone (top level) when the configuration isn't associated with a Secondary ICCID, or can be included as part of the `PrimaryConfigurations` structure.
    - `EnterpriseConfiguration`: Conditional for server-initiated ODSA
    - `MSG`: Optional for Primary ODSA

`CompanionConfiguration`, `PrimaryConfiguration` and `EnterpriseConfiguration` are multi-parameter structures that provides the configuration settings of the subscription and service running on the eSIM device.

If the eSIM profile was just activated by the Service Provider and the requesting AcquireConfiguration operation was the first one received since the activation, `CompanionConfiguration` shall contain a `DownloadInfo` element. As with the ManageSubscription operation, `DownloadInfo` specifies how to obtain the communication profile for the eSIM device from the Service Provider.

To enable eSIM profile management use cases outside of a webview and only in the case where the ECS confirms that the associated `terminal_id` does not support the webview functionality: the `CompanionConfiguration` shall contain a `CompanionDeviceInfo` element. The `CompanionDeviceInfo` shall contain all of the device related information collected by the ECS during activation. The contents of the CompanionDeviceInfo shall only be used for the purpose of eSIM profile management and discarded by the Companion ODSA application after use.

The ECS may include an `MSG` structure in order to communicate terms and conditions to the user or query information from the user without a webview.

The `CompanionConfiguration` and `PrimaryConfiguration` structures have the parameters listed in Table 43.

"AcquireConfiguration" configuration parameters	Type	Values	Description
ICCID (Conditional)	String	a valid ICCID, encoded as a 10-octet string	Integrated Circuit Card Identification - Identifier of the eSIM profile on the device's eSIM. Present if an eSIM profile exists for the device.
CompanionDeviceService (Mandatory for a Companion Configuration)	String	**SharedNumber**	Indicates that the configuration is for the "Shared Number" service (where the device carries the same MSISDN as the primary one)
		DiffNumber	Indicates that the configuration is for the "Different Number" service (where the device carries a different MSISDN from the primary one but is assigned to the same subscriber.)
		FamilyNumber	Indicates that the configuration is for the "Family Number" service (where the device carries a different MSISDN from the primary one and the MSISDN can be assigned to another individual or subscriber.)
SecondaryICCID (Conditional)	Integer	1 – True	This field is only present to indicate if the Primary Configuration was designated as a Secondary ICCID by the ECS
ServiceStatus (Mandatory)	Integer	1 to 4	Refer to Table 42 for a description of the allowed values for `ServiceStatus`.
PollingInterval (Optional)	Integer	A valid positive integer number including 0 value.	Specifies the minimum interval with which the client application may poll the ECS to refresh the current `ServiceStatus` using the AcquireConfiguration request. This parameter will be present only when `ServiceStatus`=2-ACTIVATING. If parameter is not present or value=0, this polling procedure is not triggered and ODSA App will keep waiting for any external action to continue the flow. The maximum number of `AcquireConfiguration` requests before sending a `ServiceStatus`= 4 - DEACTIVATED, NO REUSE will be defined as an ECS configuration variable (`MaxRefreshRequest`)
PollingIntervalUnit (Optional)	Integer	0 – minutes 1 – seconds 2 – deciseconds	Specifies the time unit for the PollingInterval parameter. If this parameter is not present, 0 – minutes will be considered as default value
DownloadInfo (Conditional)	Structure	multi-parameter value - see 4 for details	Specifies how and where to download the eSIM profile associated with the device. Present in case the profile is to be downloaded at this stage.

"AcquireConfiguration" configuration parameters	Type	Values	Description
CompanionDeviceInfo (Conditional)	Structure	multi-parameter value – see Table 44 for details	Includes all information collected by the ES of the companion device.
MSG (Optional)	Structure	multi-parameter value – see Table 45 for details	Includes information to be communicated and displayed to the user. Only present if the `PrimaryConfiguration` parameter is present.

Table 43. Companion and Primary Configuration for Acquire Configuration ODSA Operation

"Companion device info" informational parameters	Type	Values	Description
CompanionTerminalFriendlyName (Mandatory)	String	Any string value	User-friendly identification for the companion device which can be used by the Service Provider in Web Views.
CompanionTerminalVendor (Mandatory)	String	Any string value	Manufacturer of the companion device.
CompanionTerminalModel (Optional)	String	Any string value	Model of the companion device.
CompanionTerminalEid (Optional)	String	Value following eUICC format	eUICC identifier (EID) of the companion device being managed

Table 44. Companion and Primary Configuration for Acquire Configuration ODSA Operation

MSG parameters	Type	Values	Description
Title (Optional)	String	Any string value	The title that is displayed to the user as part of the MSG object. The client application may truncate the title for better presentation.

MSG parameters	Type	Values	Description
Message (Mandatory)	String	Any string value	The message that is displayed to the user. Please note the message may contain references to HTTP addresses (websites) that need to be highlighted and converted into links by the device/client. The Carriage Return (CR) followed by a Line Feed (LF) "\r\n" in the String represents the start of a new paragraph. The spacing for a new paragraph shall be displayed by the client application. The client application may truncate or enable scrolling of the Message for better presentation.
Accept_btn (Mandatory)	Integer	1: 'Accept' button is present. 0: 'Accept' button is absent	This indicate whether an "Accept" button is shown with the message on device UI. The action associated with the Accept button on the device/client is to clear the message box. If `Accept_btn_label` is not present the client will set the label as "Accept" or the equivalent value for the configured language of the client if it's not English.
Accept_btn_label (Optional)	String	Any string value	The label for the Accept button to be presented to the user The client application may truncate the Accept_btn_label for better presentation.
Reject_btn (Mandatory)	Integer	1: 'Decline' button is present. 0: 'Decline' button is absent.	This indicate whether a "Decline" button is shown with the message on device UI. The action associated with the Reject button on the device/client is to revert the configured services to their defined default behaviour. If `Reject_btn_label` is not present the client will set the label as "Reject" or the equivalent value for the configured language of the client if it's not English.
Reject_btn_label (Optional)	String	Any string value	The label for the Reject button to be presented to the user. The client application may truncate the Reject_btn_label for better presentation.

MSG parameters	Type	Values	Description
Accept_freetext (Mandatory)	Integer	1: A free text entry field is present. 0: A free text entry field is absent.	This indicate whether a free text entry field is shown with the message on device UI.
Accept_freetext_hint (Optional)	String	Any string value	This field may only be present if Accept_freetext is set to 1. This String is displayed in the Accept_freetext field as a hint of what value the user can enter. It is not considered an autofill so the client can display it in a different format to the text being entered in the Accept_freetext by the user. When the user enterers a value in the Accept_freetext field, this hint shall be no longer visible in the Accept_freetext. The client application may truncate the Accept_freetext_hint for better presentation.

MSG parameters	Type	Values	Description
Accept_freetext_validation (Optional)	String	Regular Expression [24]	This field may be present if Accept_freetext is set to 1. This string is a regular expression [24] that is base64 encoded and preceded by the encodedValue= prefix. This regular expression [24] shall be used to validate a match with the value that the user has entered in Accept_freetext. The client application can indicate a match or mismatch by changing the format of the entered text in Accept_freetext. The client application can also indicate a match or mismatch by changing the format of the Accept_btn. Additionally the client application may prevent the user from pressing the Accept_btn in the case of a mismatch. If Accept_freetext_validation_failed_error_text is present, the client shall display the value in Accept_freetext_validation_failed_error_text in the case of a mismatch.
Accept_freetext_validation_failed_error_text (Optional)	String	Any string value	This field may be present if Accept_freetext is set to 1 and Accept_freetext_validation is present. The client application may truncate the Accept_freetext_validation_failed_error_text for better presentation.

Table 45. Primary Configuration for Acquire Configuration ODSA Operation - MSG Information

6.5.6 AcquirePlan Operation Configuration Parameters

- Parameter names and presence:
  - `PlanOffers`: Conditional. Top level, list of all plans offered by the MNO. Present if there is one or more `PlanOffer`.
  - `PlanOffer`: Within `PlanOffers`, one or more.

The different values for the configuration parameters of the operation `AcquirePlan` are provided in Table 46

"PlanManage" configuration parameters	Type	Values	Description
PlanOffers (Conditional)	Array	Array of **`PlanOffer`** – see Table 47 for details	Array of plans offered by the MNO.

Table 46. Configuration Parameters – AcquirePlan ODSA Operation

`PlanOffer` configuration parameter is defined as a structure with several parameters as shown in Table 47.

"PlanOffer" parameters	Type	Values	Description
PlanId	String	Any string value	ID for the plan offered by the MNO.
PlanName (Optional)	String	Any string value	Name of the plan offered by the MNO. It is considered as an optional parameter due to it is not required in any request, but it is recommended to make easier the Plan identification.
PlanDescription (Optional)	String	Any string value	Description of the plan offered by the MNO. It is considered as an optional parameter due to it is not required in any request, but it is recommended to make easier the Plan identification.

Table 47. Configuration Parameters – PlanOffer for AcquirePlan

6.5.7 AcquireTemporaryToken Operation Configuration Parameters

- Parameter names and presence:
  - `TemporaryToken`: Conditional. Temporary token to allow authentication for a device that may not have the means to acquire the TOKEN.
  - `TemporaryTokenExpiry`: Conditional. Indicates the time the provided TemporaryToken expires.
  - `OperationTargets`: Conditional. The `operation_targets` associated with this temporary token and the AppID of the original AcquireTemporaryToken request.

The different values for the configuration parameters of the operation `AcquireTemporaryToken` are provided in Table 48

"AcquireTemporary Token" configuration parameters	Type	Values	Description
TemporaryToken (Conditional)	String	Any string value	This temporary token can be provided by the ECS if the ICCID supports using this token as a form of authentication for the `operation_targets` requested by the ODSA application. The temporary token can be used by a device that has no means of acquiring the TOKEN defined in this specification. To be used only for the purpose of the OperationTargets specified.
TemporaryTokenExpiry (Conditional)	Timestamp	ISO 8601 format, of the form YYYY-MM-DDThh:mm:ssTZD	This UTC value provides the expiration time for the temporary token. After the time expiration the temporary token cannot be used for authentication.
OperationTargets	String	Comma-separated list with all ODSA operations allowed to be requested using the `TemporaryToken`.	
		See table	See `operation_targets` in Table 27. The client application that will use the temporary token as its mechanism of authentication shall only make requests to the ECS with the associated target operations and the AppID used during the original `AcquireTemporaryToken` request.

Table 48. Configuration Parameters – AcquireTemporaryToken ODSA Operation

6.5.8 GetPhoneNumber Operation Configuration Parameters

- Parameter names and presence:
  - `MSISDN`: Conditional. The MSISDN of the subscription in E.164 format

"GetPhoneNumber" configuration parameters	Type	Values	Description
MSISDN (Conditional)	String	Any string value	E.164 formatted phone number It is possible to provide the base64 encoding of the value by preceding it with `encodedValue=`

Table 49. Configuration Parameters – GetPhoneNumber ODSA Operation

6.5.9 Client Processing of Parameters Associated with SP Web Portal

The response to `CheckEligibility` and `ManageSubscription` operations may contain response parameters that permit a client application to interact with a Service Provider's portal web server. This clause explains how the client application should process those portal-related parameters.

For `CheckEligibility` the response parameters associated with an SP web portal are:

- NotEnabled**URL**
- NotEnabled**UserData**
- NotEnabled**ContentsType**
- GeneralError**URL**
- GeneralError**UserData**

For `ManageSubscription` the response parameters associated with a SP web portal are:

- SubscriptionService**URL**
- SubscriptionService**UserData**
- SubscriptionService**ContentsType**

Refer to 6.5.2 and 6.5.3 for the definition of those response parameters. For simplicity purposes, this clause refers to the parameters by their common endings: `URL`, `UserData` and `ContentTypes`.

The `URL` parameter specifies the web address of the SP portal. The device client connects to the portal by sending a GET or POST request to `URL`. `ContentsType` specifies the format of `UserData` and how the resulting HTTP request should carry `UserData` to the SP web portal.

An overview of the procedure for `ManageSubscription` is shown in the Figure 18.

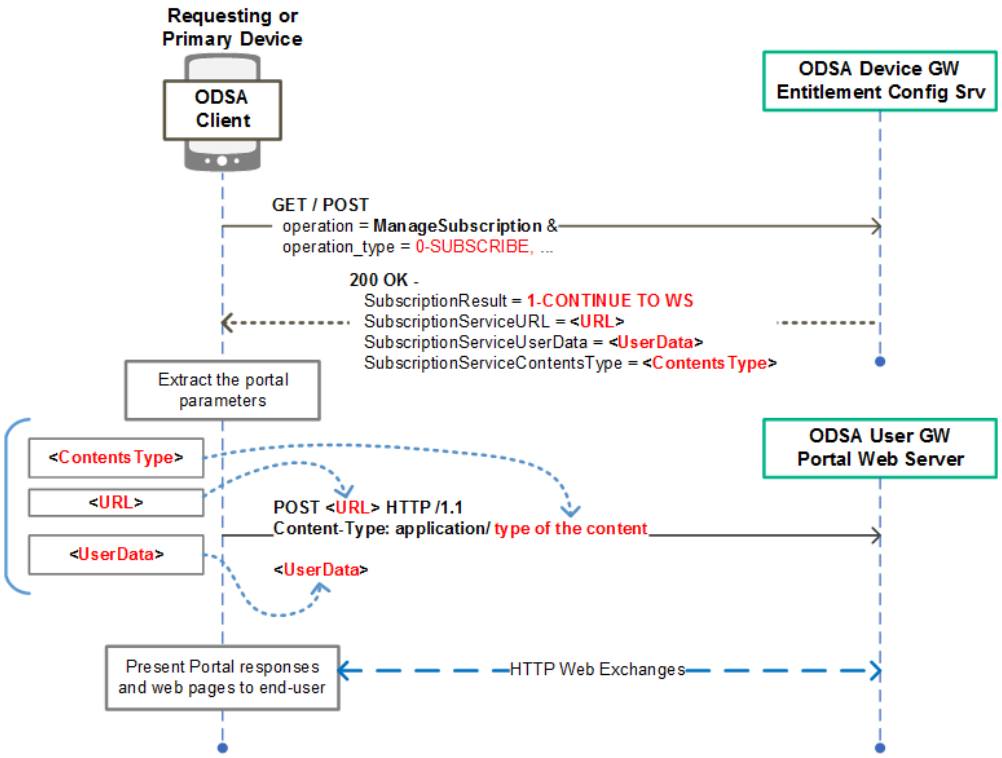

Figure 18. Example Processing of Web Portal Response Parameters by Client

The processing rules for `UserData` and `ContentTypes` are provided in Table 50.

ContentTypes parameter	UserData parameter	Expected HTTP Request to SP Web Portal
NOT present	Contains user information as query parameters, of the form `field1=value1&field=value2&...` to be included in GET request.	`GET <URL>?<UserData> HTTP /1.1` `. . .`

value of `json`	Contains user information presented in a JSON object value.
	If it is preceded by `encodedValue=`, `UserData` is a base64 string and must first be decoded to text before inclusion in POST.
		`POST <URL> HTTP /1.1`
		`Content-Type: application/json`
		`. . .`
		
		`<UserData as JSON object>`

value of `xml`	Contains user information presented in an XML document.
	If it is preceded by `encodedValue=`, `UserData` is a base64 string and must first be decoded to text before inclusion in POST.
		`POST <URL> HTTP /1.1`
		`Content-Type:`
		`text/xml,application/xml`
		`. . .`
		
		`<UserData as XML document>`

Table 50. Processing Rules for the Response Parameters ContentTypes and UserData

6.5.10 VerifyPhoneNumber Operation Configuration Parameters

Parameter names and presence:

- `PhoneNumberVerification`: Mandatory. Indicates if the MSISDNs match.
- `msisdn`: Optional. The MSISDN of the subscription in E.164 format

"VerifyPhoneNumber" configuration parameters	Type	Values	Description
PhoneNumberVerification	Integer	Indicates the result of the Phone Number verification	
		0 – FAILURE	MSISDNs don't match
		1 – SUCCESS	MSISDNs match
msisdn (Optional)	String	E.164 formatted phone number	This parameter could be present when SUCCESS. If present, it indicates the MSISDN (the one from the request) that has been verified successfully.

Table 51. Configuration Parameters - VerifyPhoneNumber Operation

6.5.11 GetSubscriberInfo Operation Configuration Parameters

Parameter names and presence:

- `SubscriberInfo`: Conditional. Application specific subscriber information

The different values for the configuration parameters of the operation `GetSubscriberInfo` are provided in Table 52

"GetSubscriberInfo" configuration parameters	Type	Values	Description
SubscriberInfo (Conditional)	Structure	multi-parameter value - see Table 53 for details	Subscriber information details.

Table 52. Configuration Parameters – GetSubscriberInfo Operation

"SubscriberInfo" configuration parameters for ap2014	Type	Values	Description
MSISDN	String	Any string value	E.164 formatted phone number

"SubscriberInfo" configuration parameters for ap2014	Type	Values	Description
SimIdType	Integer	0 – IMSI 1 – UUID 2 – IMSI HASH	Specifies the type of unique identifier used in SimID parameter
SimID	String	Any string value	For SimIdType=0 ("IMSI"): International Mobile Subscriber Identity as per ITU E.212 or 3GPP TS 23.003 standards. For SimIdType=1 ("UUID") For SimIdType=2 ("IMSI HASH"): "keyed hashing HMAC SHA256" where the key is owned by the MNO" (recommendation)
MvnoName (Optional)	String	Any string value	Applicable for MVNO-specific features, specifies the MVNO name to which the subscriber belongs. It can be Gid1/2 or a unique name.

Table 53. Configuration Parameters – SubscriberInfo for ap2014

6.6 Examples of ODSA Responses

6.6.1 CheckEligibility Response Example

Table 54 presents an example for the CheckEligibility response to a Companion ODSA application.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS">
       <parm name="version" value="1"/>
       <parm name="validity" value="172800"/>
    </characteristic>

    <characteristic type="TOKEN">
      <parm name="token" value="ASH127AHHA88SF"/>
    </characteristic>

    <characteristic type="APPLICATION">
      <parm name="AppID" value="ap2006"/>
      <parm name="CompanionAppEligibility" value="1"/>
      <parm name="CompanionDeviceServices" value="SharedNumber"/>
      <parm name="NotEnabledURL" value="http://www.MNO.org/AppNotAllowed"/>
      <parm name="NotEnabledUserData" value="msisdn=XX&amp;device_id=XX"/>
      <parm name="OperationResult" value="1"/>
    </characteristic>
</wap-provisioningdoc>
```

Table 54. Example of a CheckEligibility ODSA Response in XML format

Table 55 presents an example for the `CheckEligibility` response to a Companion ODSA application in JSON format.

```
{
    "Vers" : {
        "version" : "1",
        "validity" : "172800"
    },
    "Token" : {              // Optional
        "token" : "ASH127AHHA88SF"
    },
    "ap2006" : {            // ODSA for Companion Device app
        "CompanionAppEligibility" : "1",
        "CompanionDeviceServices" : "SharedNumber",
        "NotEnabledURL" : "http://www.MNO.org/AppNotAllowed",
        "NotEnabledUserData" : "msisdn=XX&amp;device_id=XX",
        "OperationResult" : "1"
    }
}
```

Table 55. Example of a CheckEligibility ODSA Response in JSON format

6.6.2 ManageService Response Example

Table 56 presents an example for the `ManageService` response to a Companion ODSA application.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS">
       <parm name="version" value="1"/>
       <parm name="validity" value="172800"/>
    </characteristic>

    <characteristic type="TOKEN">
      <parm name="token" value="ASH127AHHA88SF"/>
    </characteristic>

    <characteristic type="APPLICATION">
      <parm name="AppID" value="ap2006"/>
      <parm name="ServiceStatus" value="3"/>
      <parm name="OperationResult" value="1"/>
    </characteristic>
</wap-provisioningdoc>
```

Table 56. Example of a ManageService ODSA Response

Table 57 presents an example for the `ManageService` response to a Companion ODSA application in JSON format.

```
{
    "Vers" : {
        "version" : "1",
        "validity" : "172800"
    },
    "Token" : {               // Optional
        "token" : "ASH127AHHA88SF"
    },
    "ap2006" : {             // ODSA for Companion Device app
        "ServiceStatus" : "3",
        "OperationResult" : "1"
    }
}
```

Table 57. Example of a ManageService ODSA Response in JSON format

6.6.3 ManageSubscription Response Example

Table 58 presents an example for the `ManageSubscription` response in XML format to a Companion or Primary ODSA application. This response indicates that the end-user is to be sent to an ODSA portal web server.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS"
       <parm name="version" value="1"/>
       <parm name="validity" value="172800"/>
    </characteristic>

    <characteristic type="TOKEN">
      <parm name="token" value="ASH127AHHA88SF"/>
    </characteristic>

    <characteristic type="APPLICATION">
      <parm name="AppID" value="ap2006"/>
      <parm name="SubscriptionServiceURL" value="http://www.MNO.org/CDSubs"/>
      <parm name="SubscriptionServiceUserData" value="imsi=XX&amp;msisdn=XX"/>
      <parm name="SubscriptionResult" value="1"/>  <!-- continue to websheet  -->
      <parm name="OperationResult" value="1"/>
    </characteristic>
</wap-provisioningdoc>
```

Table 58. Example of a ManageSubscription ODSA Response in XML format to send user to ODSA portal.

Table 59 presents an example for the `ManageSubscription` response in XML format to a Companion or Primary ODSA application. This response provides information on the eSIM profile to download.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS"
        <parm name="version" value="1"/>
        <parm name="validity" value="172800"/>
    </characteristic>

    <characteristic type="TOKEN">
        <parm name="token" value="ASH127AHHA88SF"/>
    </characteristic>

    <characteristic type="APPLICATION">
        <parm name="AppID" value="ap2006"/>
        <characteristic type="DownloadInfo">
            <parm name="ProfileIccid" value="1111111111111111"/>
            <parm name="ProfileSmdpAddress" value="SMDP+ ADDR"/>
        </characteristic>
        <parm name="SubscriptionResult" value="2"/>      <!—download profile  -->
        <parm name="OperationResult" value="1"/>
    </characteristic>
</wap-provisioningdoc>
```

Table 59. Example of a ManageSubscription ODSA Response in XML format with profile download information.

Table 60 presents an example for the `ManageSubscription` response in JSON format to a Companion or Primary ODSA application. This response indicates that the end-user is to be sent to an ODSA portal web server.

```
{
    "Vers" : {
        "version" : "1",
        "validity" : "172800"
    },
    "Token" : {              // Optional
        "token" : "ASH127AHHA88SF"
    },
    "ap2006" : {            // ODSA for Companion Device app
        "SubscriptionServiceURL" : "http://www.MNO.org/CDSubs",
        "SubscriptionServiceUserData" : "imsi=XX&amp;msisdn=XX",
        "SubscriptionResult" : "1",               // continue to websheet
        "OperationResult" : "1"
    }
}
```

Table 60. Example of a ManageSubscription ODSA Response in JSON format to send user to ODSA portal.

Table 61 presents an example for the `ManageSubscription` response in JSON format to a Companion or Primary ODSA application. This response provides information on the eSIM profile to download.

```
{
    "Vers" : {
        "version" : "1",
        "validity" : "172800"
    },
    "Token" : {              // Optional
        "token" : "ASH127AHHA88SF"
    },
    "ap2006" : {            // ODSA for Companion Device app
        "DownloadInfo" : {
            "SubscriptionServiceURL" : "SMDP+ ADDR",
            "ProfileActivationCode" : "COMM PROFILE CODE"
        },
        "SubscriptionResult" : "2",                // download profile
        "OperationResult" : "1"
    }
}
```

Table 61. Example of a ManageSubscription ODSA Response in JSON format with profile download information.

6.6.4 AcquireConfiguration Response Example

Table 62 presents an example for the `AcquireConfiguration` operation in XML format for a Companion ODSA application.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS">
        <parm name="version" value="1"/>
        <parm name="validity" value="172800"/>
    </characteristic>
    <characteristic type="TOKEN">
        <parm name="token" value="ASH127AHHA88SF"/>
    </characteristic>
    <characteristic type="APPLICATION">
        <parm name="AppID" value="ap2006"/>
        <characteristic type="CompanionConfigurations">
            <characteristic type="CompanionConfiguration">
                <parm name="ICCID" value="8991101200003204510"/>
                <parm name="CompanionDeviceService" value="SharedNumber"/>
                <parm name="ServiceStatus" value="1"/>
            </characteristic>
        </characteristic>
        <parm name="OperationResult" value="1"/>
    </characteristic>
</wap-provisioningdoc>
```

Table 62. Example of an AcquireConfiguration ODSA Response in XML format

Table 63 presents an example for the `AcquireConfiguration` operation in JSON format for a Companion ODSA application.

```
{
    "Vers" : {
        "version" : "1",
        "validity" : "172800"
    },
    "Token" : {              // Optional
        "token" : "ASH127AHHA88SF"
    },
    "ap2006" : {            // ODSA for Companion Device app
        "CompanionConfigurations" : [{
         "CompanionConfiguration" : {
            "ICCID" : "8991101200003204510",
            "CompanionDeviceService" : "SharedNumber",
            "ServiceStatus" : "1"
        }
    }],
    "OperationResult" : "1"
    }
}
```

Table 63. Example of an AcquireConfiguration ODSA Response in JSON format

Table 64 presents an example for the `AcquireConfiguration` operation in XML format for a Companion ODSA application.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS">
        <parm name="version" value="1"/>
        <parm name="validity" value="172800"/>
    </characteristic>
    <characteristic type="TOKEN">
        <parm name="token" value="ASH127AHHA88SF"/>
    </characteristic>
    <characteristic type="APPLICATION">
        <parm name="AppID" value="ap2009"/>
        <characteristic type="PrimaryConfigurations">
            <characteristic type="PrimaryConfiguration">
                <parm name="ICCID" value="8991101200003204510"/>
                <parm name="ServiceStatus" value="1"/>
            </characteristic>
            <characteristic type="PrimaryConfiguration">
                <parm name="ICCID" value="8991101200003204514"/>
                <parm name="ServiceStatus" value="4"/>
                <parm name="SecondaryICCID" value="1"/>
            </characteristic>
        </characteristic>
        <parm name="OperationResult" value="1"/>
    </characteristic>
</wap-provisioningdoc>
```

Table 64. Example of an AcquireConfiguration ODSA Response in XML format

Table 65 presents an example for the `AcquireConfiguration` operation in JSON format for a Companion ODSA application.

```
{
    "Vers" : {
        "version" : "1",
        "validity" : "172800"
    },
    "Token" : {              // Optional
        "token" : "ASH127AHHA88SF"
    },
    "ap2009" : {         // ODSA for Primary with Multiple Primary configurations
        "PrimaryConfigurations" : [{
         "PrimaryConfiguration" : {
            "ICCID" : "8991101200003204510",
            "ServiceStatus" : "1",
        },
         "PrimaryConfiguration" : {
            "ICCID" : "8991101200003204514",
            "ServiceStatus" : "4",
            "SecondaryICCID" : "1"
        }
    }],
    "OperationResult" : "1"
    }
}
```

Table 65. Example of an AcquireConfiguration ODSA Response in JSON format

6.6.5 AcquirePlan Response Example

Table 66 presents an example for the `AcquirePlan` operation in XML format for a Server-initiated ODSA application.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS">
        <parm name="version" value="1"/>
        <parm name="validity" value="172800"/>
    </characteristic>

    <characteristic type="TOKEN">
        <parm name="token" value="ASH127AHHA88SF"/>
    </characteristic>

    <characteristic type="APPLICATION">
        <parm name="AppID" value="ap2011"/>
        <characteristic type="PlanOffers">
            <characteristic type="PlanOffer">
                <parm name="PlanId" value="Plan0001"/>
                <parm name="PlanName" value="Family Plan"/>
                <parm name="PlanDescription" value="This is the description of
the Plan0001"/>
            </characteristic>
            <characteristic type="PlanOffer">
                <parm name="PlanId" value="Plan0376/>
                <parm name="PlanName" value="All included Plan"/>
                <parm name="PlanDescription" value="This is the description of
the Plan0376"/>
            </characteristic>
        </characteristic>
        <parm name="OperationResult" value="1"/>
    </characteristic>
</wap-provisioningdoc>
```

Table 66. Example of an AcquirePlan Server-initiated ODSA Response in XML format

Table 67 presents an example for the `AcquirePlan` operation in XML format for a Server-initiated ODSA application.

```
{
    "Vers" : {
        "version" : "1",
        "validity" : "172800"
    },
    "Token" : {              // Optional
        "token" : "ASH127AHHA88SF"
    },
    "ap2011" : {            // ODSA for Server-initiated app
        "PlanOffers" : [{
            "PlanOffer" : {
                  "PlanId" : " Plan0001",
                  "PlanName" : "Family Plan",
                  "PlanDescription" : "This is the description of the Plan0001"
             },
      },{
             "PlanOffer" : {
                  "PlanId " : "Plan0376",
                  "PlanName " : "All included Plan",
                  "PlanDescription" : "This is the description of the Plan0376"
             }
        }],
        "OperationResult" : "1"
    }
}
```

Table 67. Example of a AcquirePlan Server-initiated ODSA Response in JSON format

6.6.6 AcquireTemporaryToken Response Example

Table 68 presents an example for the AcquireTemporaryToken response in XML format to a Primary ODSA application. This response provides the ODSA application with the `TemporaryToken` to be used for an eSIM transfer.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS"
       <parm name="version" value="1"/>
       <parm name="validity" value="172800"/>
    </characteristic>

    <characteristic type="TOKEN">
      <parm name="token" value="ASH127AHHA88SF"/>
    </characteristic>

    <characteristic type="APPLICATION">
      <parm name="AppID" value="ap2009"/>
      <parm name="TemporaryToken" value="A8daAd8ads7fau34789947kjhsfad;kjfh"/>
      <parm name="TemporaryTokenExpiry" value="2019-01-29T13:15:31-08:00"/>
      <parm name="OperationTargets"
value="ManageSubscription,AcquireConfiguration”/>
      <parm name="OperationResult" value="1"/>
    </characteristic>
</wap-provisioningdoc>
```

Table 68. Example of an AcquireTemporaryToken Response in XML

6.6.7 GetPhoneNumber Response Example

Table 69 presents an example for GetPhoneNumber response in XML.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS"
       <parm name="version" value="1"/>
       <parm name="validity" value="172800"/>
    </characteristic>

    <characteristic type="TOKEN">
      <parm name="token" value="ASH127AHHA88SF"/>
    </characteristic>

    <characteristic type="APPLICATION">
      <parm name="AppID" value=”ap2014"/>
      <parm name="MSISDN" value="+14058885769"/>
    </characteristic>
</wap-provisioningdoc>
```

Table 69. Example of a GetPhoneNumber Response in XML

6.6.8 VerifyPhoneNumber Response Example

Table 70 presents an example for `VerifyPhoneNumber` response in XML

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS"
       <parm name="version" value="1"/>
       <parm name="validity" value="172800"/>
    </characteristic>

    <characteristic type="TOKEN">
      <parm name="token" value="ASH127AHHA88SF"/>
    </characteristic>

    <characteristic type="APPLICATION">
      <parm name="AppID" value=”ap2014"/>
      <parm name="OperationResult" value=”1"/>
      <parm name="PhoneNumberVerification" value="1"/>
      <parm name="msisdn" value="+14058885769"/>  //Optional
    </characteristic>
</wap-provisioningdoc>
```

Table 70. Example of a VerifyPhoneNumber Response in XML

6.6.9 GetSubscriberInfo Response Example

Table 71 presents an example for the `GetSubscriberInfo` response in XML.

```
<?xml version=”1.0”?>
<wap-provisioningdoc version=”1.1”>
    <characteristic type=”VERS”
       <parm name=”version” value=”1”/>
       <parm name="validity" value="172800"/>
    </characteristic>

    <characteristic type="APPLICATION">
      <parm name="AppID" value=”ap2014"/>
      <characteristic type="SubscriberInfo">
        <parm name="MSISDN" value="+14058885769"/>
        <parm name="SimIdType" value="2"/>
        <parm name="SimID"
value="ffc72d247a9c60d3220020b62bca7cfd0ea9e159076370586944968de219080a"/>
        <parm name="MvnoName" value="MVNO_222"/>
      </characteristic>
    </characteristic>
</wap-provisioningdoc>
```

Table 71. Example of a GetSubscriberInfo Response in XML format

6.7 ODSA Application Considerations around Web View Callback

During the procedure for ODSA on Companion or Primary eSIM devices, end-users can be presented with a set of web views specific to the Operator. The web views are hosted by an Operator portal web server as shown in Figure 10.

To support proper communication between web views and the ODSA application, the application should support JS callbacks to allow for the portal to share the following events and corresponding data elements described in Table 72.

Callback Event	Data	Description
Communication profile ready for download	Profile download method and corresponding parameters (Activation Code or SM-DP+ address, see Table 40 for details)	The eSIM ODSA procedure was a success. The resulting communication profile can be downloaded.
Web flow finished	None	The end-user has completed the ODSA web view flow. The device app needs to perform an `AcquireConfiguration` operation to retrieve the status of the eSIM profile and associated service.
Web flow dismissed	None	The end-user or web portal logic has ended the ODSA web views without completing the ODSA procedure. An eSIM profile is not available.
End-user logged out	None	The end-user was logged out of the web views. The active authentication token must be deleted, and re-authentication is required for subsequent requests.

Table 72. Callback Events for ODSA Web Views

The different callback functions are embedded in the **ODSAServiceFlow** object. They are defined to reflect the state of the web logic according to the opened web view:

Callback name	Webview opened on SubscriptionServiceUrl	Webview opened on NotEnabledUrl	Webview opened on GeneralErrorURL
profileReadyWithActivationCode (activationCode)	X		
profileReadyWithDefaultSmdp (defaultSmdpAddress, iccid = "0")	X		
SelectionCompleted (ICCID, IMEI)	X		
finishFlow (next_action)	X		
dismissFlow ()	X	X	X
deleteToken()	X		
deleteProfileInUse()	X		
checkProfileServiceStatus ()	X		

Table 73. Callback signatures for ODSA Web Views

6.7.1 profileReadyWithActivationCode(activationCode)

Calling this method indicates that an eSIM profile, identified by the activation code, is ready for download.

The parameter `activationCode` is mandatory. It is a string with GSMA SGP .22 v2.1 or higher format.

After this call back is called, the related eSIM profile will be downloaded, and the web view will not be closed.

6.7.2 profileReadyWithDefaultSmdp(defaultSmdpAddress, iccid)

Calling this method indicates that an eSIM profile, identified by its iccid, from a SM-DP+ server, is ready for download.

Default Smdp here does not refer to an SM-DP+ being the Default SM-DP+ server for the requesting device, but to the eSIM profile being prepared for Default SM-DP+ Download Use case, as defined in GSMA SGP.22 v2.1 or higher.

The parameter `defaultSmdpAddress` is mandatory, it is a string containing the FQDN of the SM-DP+, not an URL.

The parameter iccid is a string of the ICCID to be downloaded.

After this call back is called, the related eSIM profile will be downloaded, and the web view will not be closed.

6.7.3 SelectionCompleted(iccid, imei) callback function

Calling this method indicates that an eSIM profile, identified by its old ICCID and/or IMEI, was selected by the user on the Websheet.

The parameter `iccid` is a string, whose default value is empty.

The parameter `imei` is a string, whose default value is empty.

After this callback, the webview will be closed.

6.7.4 dismissFlow() callback function

Calling this method ends prematurely the ODSA service flow, whatever the cause (user action, user not eligible…), without a service profile being downloaded.

This callback has no parameter.

The web view to the end-user will be closed.

The call flows in the next figures show some examples of the callback use in the different webviews.

In the Figure 19, the webview is opened in step 10, following an end-user action. While the subscription page is displayed (13), the end-user may cancel the subscription, for instance with a dedicated button on the page. This should call the **dismissFlow()** callback. The ODSA client closes the webview.

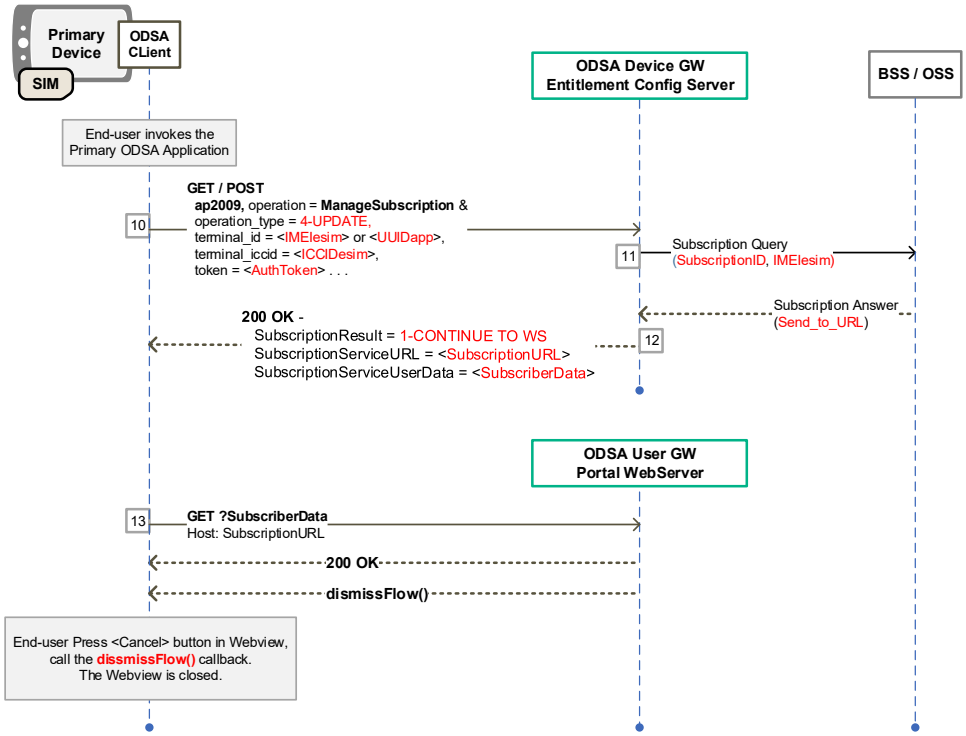

Figure 19. Example of dismissFlow callback in SubscriptionServiceURL webview

In the Figure 20, the webview is opened in step 10, following an end-user action. Once the "not enabled" page is displayed (13), giving information about the cause of the ineligibility, the end-user may discard it, for instance with a "close" button on the page. This should call the dismissFlow() callback. The ODSA client closes the webview.

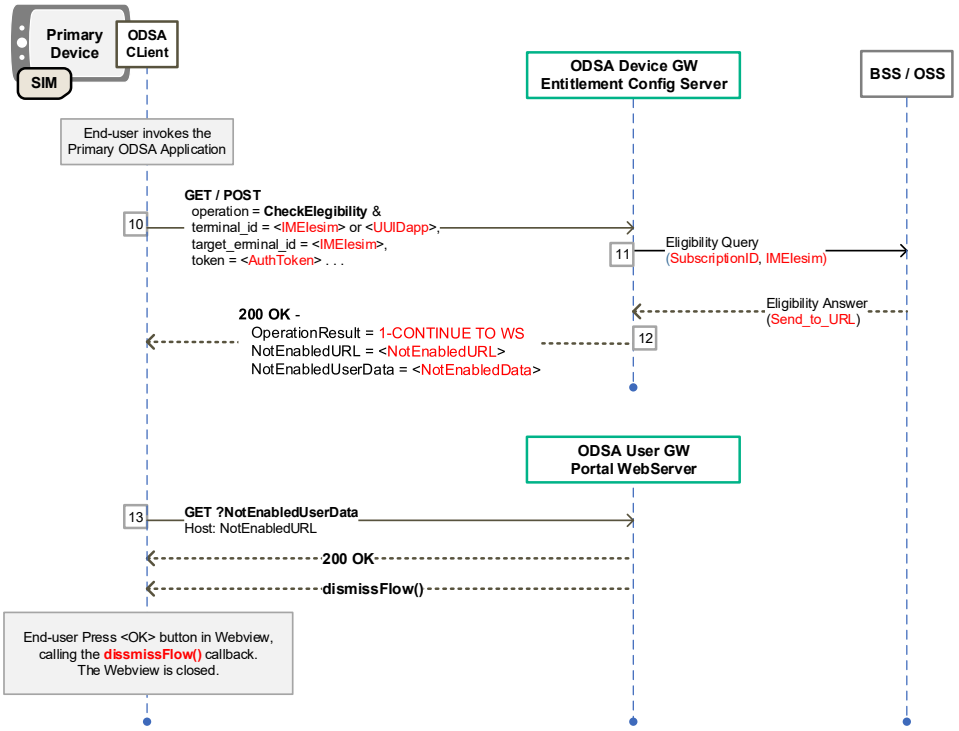

Figure 20. Example of dismissFlow callback in NotEnabledURL webview

6.7.5 finishFlow(next_action(optional))

Calling this method shall dismiss the ODSA Web Service Flow on device side and trigger the `next_action` request to the ECS.

This callback could include a parameter (`next_action`). This parameter is defined as a String. If the callback doesn't contain any parameter, the `AcquireConfiguration` value will be considered as the action to be triggered.

The web view will be closed.

6.7.6 deleteToken()

Calling this method erases the current authentication token to perform a full re-authentication request. This may be called in the subscription webview when the user account has been changed, for instance.

This callback has no parameter.

6.7.7 checkProfileServiceStatus()

Calling this method triggers the client to check the `ServiceStatus` using an AcquireConfiguration request without the need of for the Web View to dismiss the ODSA Web Service Flow.

This callback has no parameter.

6.7.8 deleteProfileInUse(iccid, msisdn (optional))

Calling this method notifies that the user needs to delete the profile having the ICCID in the parameter to complete the subscription transfer. This may be called in the subscription webview when the profile in use needs to be deleted before the subscription is transferred.

6.8 Void

6.9 Information Representation for On-Device Service Activation (ODSA) Entitlement and Configuration

This section presents an information representation that can be used to trigger the relevant ODSA procedure by the ODSA client application. The information shall be coded as a concatenation of the string listed in the Table 74 using a URI format as defined in RFC3986 [22].

Name	Description
Scheme	Shall be set to "esim".
Delimiter	Shall be set to ":".
Path	Expected operation for the ODSA client application receiving this information.
Delimiter	Shall be present and set to "?" if any of the following query components is present.
Query	Additional information for the expected operation in the form of "key=value" pairs.

Table 74. Device Information Format

Each key/value pair in the query part shall be concatenated by using "&" as a delimiter if there are more than one key/value pair in the query component. Any of keys can be present in any order.

6.9.1 Device Information for Subscription Transfer

In order to prepare an appropriate eSIM profile during the subscription transfer, the Primary ODSA client application on the old device may need to provide to the ECS relevant information of the new device where the prepared eSIM profile will be installed.

This section presents a device information representation for subscription transfer. The information shall contain the path set to "transfer" when this device information is used in the context of the subscription transfer starting from the old device. Each key/value pair for subscription transfer is defined in Table 75.

Key	Value
eid	EID of the eUICC, the numeric text representation SHALL comprise 32 digits, where each digit is represented by one character in the set [0123456789].
imei	IMEI of the Device, the numeric text representation SHALL comprise 15 digits, where each digit is represented by one character in the set [0123456789].

Table 75. Query Component for Device Information

The device information can be represented in a text string restricted to Byte mode character set defined in table 6 of ISO/IEC 18004:2015 [23] and the equivalent QR code according to ISO/IEC 18004:2015 [23]. Examples of the device information representation are as follows:

- esim:transfer?eid=8900156701020304050607080910115 2
  (if an EID is present)
- esim:transfer?imei=351234510000011
  (if an IMEI is present)
- esim:transfer?eid=8900112233445566778899001122334 4&imei=351234520000029
  (if an EID and IMEI are present)

6.9.2 Service Provider related information for eSIM subscription activation via ODSA

The Primary ODSA client may need relevant information of the Service Provider in order to trigger appropriate procedure for new eSIM subscription activation via ODSA.

This section presents a Service Provider related information for new eSIM subscription activation. The information shall contain the path set to "newsubscription" when this information is used for new eSIM subscription activation. Each key/value pair in the query for eSIM subscription activation is defined in Table 75a.

Key	Value
mcc	Mobile Country Code of the home network in decimal format to a 3-digits.
mnc	Mobile Network Code of the home network in decimal format and with a 2-digit MNC padded out to 3 digits by inserting a 0 at the beginning.
gid1	Group ID Level 1 in string. This is referring to content of EF GID1 (file identifier ‘6F3E’) and coded in 3GPP TS 31.102 [21].
gid2	Group ID Level 2 in string. This is referring to content of EF GID2 (file identifier '6F3F') and coded in 3GPP TS 31.102 [21].
esurl	The FQDN (Fully Qualified Domain Name) of the ECS the client application can send requests to. It is in UTF8String format.

Table 75a. Query Component for Device Information

The information can be represented in a text string restricted to Byte mode character set defined in table 6 of ISO/IEC 18004:2015 [23] and the equivalent QR code according to ISO/IEC 18004:2015 [23]. Examples of the information representation are as follows:

- esim:newsubscription?mcc=123&mnc=056&gid1=ABC0000000000000
  (if an mcc, mnc and gid1 are present)
- esim:newsubscription?mcc=124&mnc=789&gid1=BA01230000000000&gid2=B30000 0000000000
  (if an an mcc, mnc, gid1 and gid2 are present)
- esim:newsubscription?esurl=ECS.EXAMPLE.COM
  (if an esurl is present)

7 Companion ODSA Procedure Call Flows

The following sections present a number of informational call flows for the different user experiences and use cases of the Companion ODSA procedure. The ODSA client application on the requesting device is invoked at the request of the end-user and should capture proper user consent to have access to the companion device.

The exchanges between the Entitlement Configuration Server (ECS) (aka ODSA Device Gateway) and the Service Provider's (SP) back-end systems are shown for informational purposes only. This applies as well for the exchanges that involve the ODSA Portal Web Server.

7.1 Subscription Activation via ODSA Portal – Initial Steps

The following presents the case where:

- The companion ODSA client application is allowed for the type of requesting device and enabled for the end-user (entitled).
- The companion device does not have an active eSIM/subscription from the Service Provider.
- The SP's ODSA portal web server is responsible for completing the subscription activation for the companion device.

Figure 21 shows the initial steps of the flow involving the SP's ODSA portal, where the Companion ODSA client application acquires proper entitlement and subscription data from the SP's ECS. The steps are:

1. End-user invokes the Companion ODSA client application on the requesting device which connects with the companion device to initiate the ODSA procedure (over a protocol outside the scope of this specification).
2. The companion ODSA client application makes a **CheckEligibility** request to the ECS.
3. The ECS queries the SP's back-end system managing the end-user's entitlements and services.
4. The ECS processes the answer from the SP's back-end system and generates the proper 200 OK response containing `CompanionDevice` entitlement set to ENABLED and allowed services in the `CompanionDeviceServices` field set to **`SharedNumber`**.
5. Since the `CompanionDevice` entitlement value is correct and target service is allowed, the companion ODSA client application sends an **AcquireConfiguration** request to the ECS to obtain information on any eSIM profiles associated with the companion device.
6. The ECS queries the SP's back-end system managing the subscriptions and active eSIM profiles. The device may also add the parameters `notif_token` and `notif_action` to the **AcquireConfiguration** request in case Infrastructure-based push-notifications (see 2.6.2) should be used later. These parameters may be added to any GET/POST request by the device.
7. The ECS processes the response from the SP's back-end system and generates the proper 200 OK response containing `CompanionDeviceConfigurations` without

any `CompanionConfiguration` (no eSIM profile/subscription is associated with the companion device). If in step 5, the device registered for push notifications, the ECS now also uses the `RegisterNotifStatus` parameter to notify the device about the Notification Registration (see 2.9.5).

8. The companion ODSA client application makes a **ManageSubscription** request to the ECS with an `operation_type` set to SUBSCRIBE (value of 0) to initiate the subscription procedure for the companion device.
9. The ECS queries the SP's back-end system to determine the next step and method to use for the companion device's subscription request.
10. The ECS processes the response from the SP's back-end system and generates the proper 200 OK response to send the application and end-user to the SP's ODSA portal. The response contains a `SubscriptionResult` set to CONTINUE_TO_WS (value of 1), and `SubscriptionServiceURL` along with `SubscriptionServiceUserData` presenting the URL of the ODSA Portal web server and any user-specific data that would be useful to the Portal.

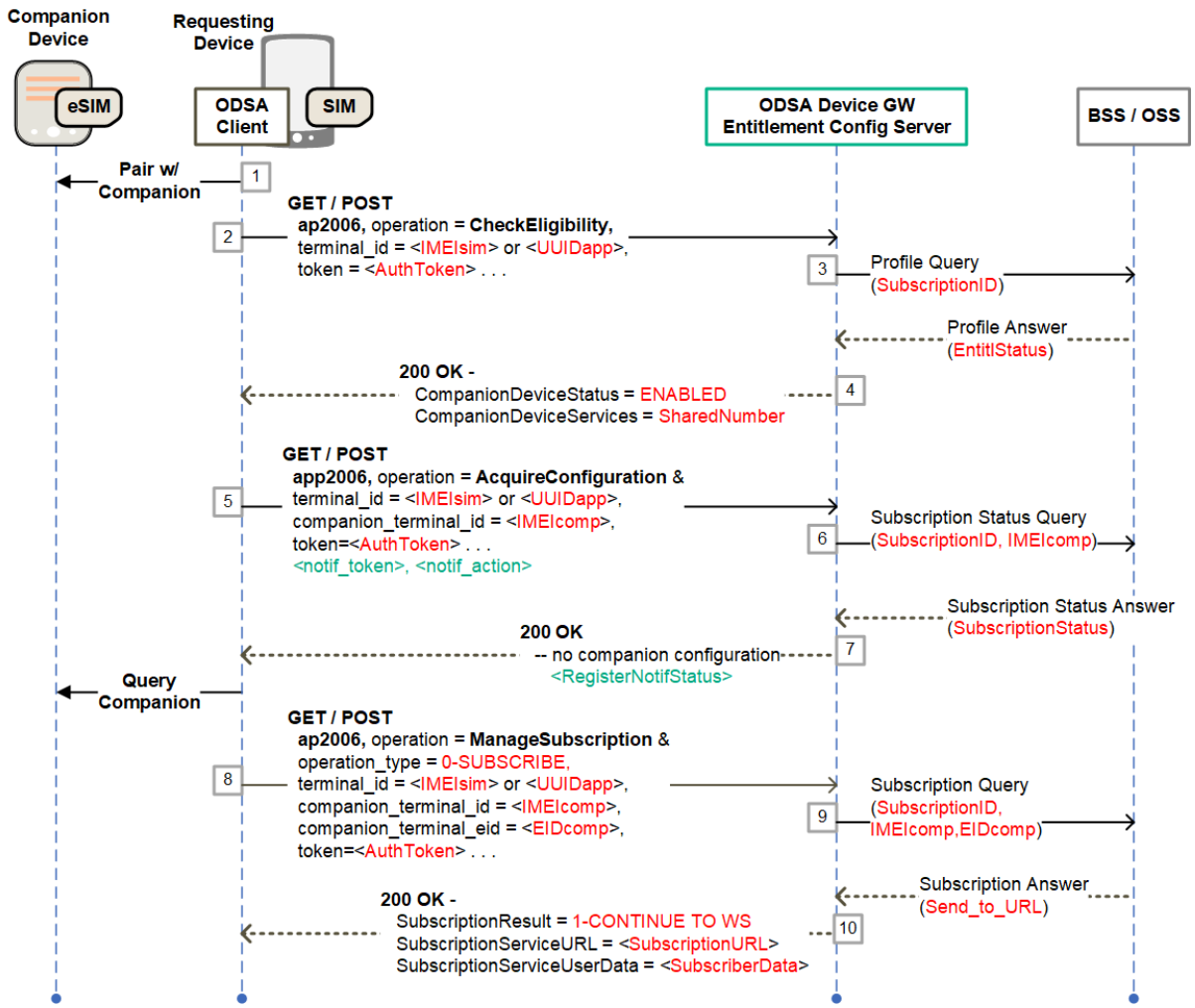

Figure 21. Initial steps for companion ODSA procedure involving ODSA portal.

7.2 ODSA Portal with Immediate Download Info – Final Steps

The following presents the case where:

- The companion ODSA client application was already informed to use the SP's ODSA portal to complete the subscription procedure (refer to 7.1).
- The ODSA portal is able to generate the eSIM profile download information as a result of the exchanges with the end-user.

Figure 22 shows the final steps of the Companion ODSA procedure in the case where the ODSA portal provides the eSIM profile download information back to the application (immediate delivery). The steps are:

11. The ODSA client application connects with the ODSA portal web server using the URL provided in the **ManageSubscription** operation response, allowing the web pages from the portal to be displayed to the end-user.
12. The ODSA portal web server presents a set of plan offers to the end-user and captures the selection from the end-user.
13. The ODSA portal makes a request towards the SP's back-end system to activate the selected plan and subscription.
14. The SP's back-end system interacts with the SM-DP+ over the ES2+ interface to make the required eSIM profile requests associated with the new subscription (for example, `DownloadOrder`, `ConfirmOrder` and `ReleaseProfile`) resulting in an activation code and ICCID for the companion device.
15. The ODSA portal provides the communication eSIM profile download information (activation code) to the ODSA client application using a JavaScript call back function.
16. The ODSA client application informs the companion device to download the eSIM profile.
17. The companion device downloads the eSIM profile from the SM-DP+
18. Optional - The ODSA application makes a **ManageService** request to the ECS with an `operation_type` set to ACTIVATE SERVICE (value of 10) to have the network activate and provision the `NumberShare` service on the companion device.
19. The ECS makes the appropriate requests to the SP's back-end system for service activation on the companion device's subscription.
20. The SP's back-end system replies back with service status and the ECS generates the proper response with service status to the ODSA client application.
21. The ODSA client application makes an **AcquireConfiguration** request to the ECS to verify that the subscription and service for the companion device are in the proper states.
22. The ECS queries the SP's back-end system managing the subscriptions and profiles.
23. The ECS processes the response from the SP's back-end system and generates the proper 200 OK response containing `CompanionDeviceConfigurations` with a `CompanionDeviceConfiguration` entry for the newly active subscription bearing the ACTIVATED status (value of 1).
24. As the companion device’s subscription and service are in the right state, the ODSA client application informs the companion device to initiate cellular service.

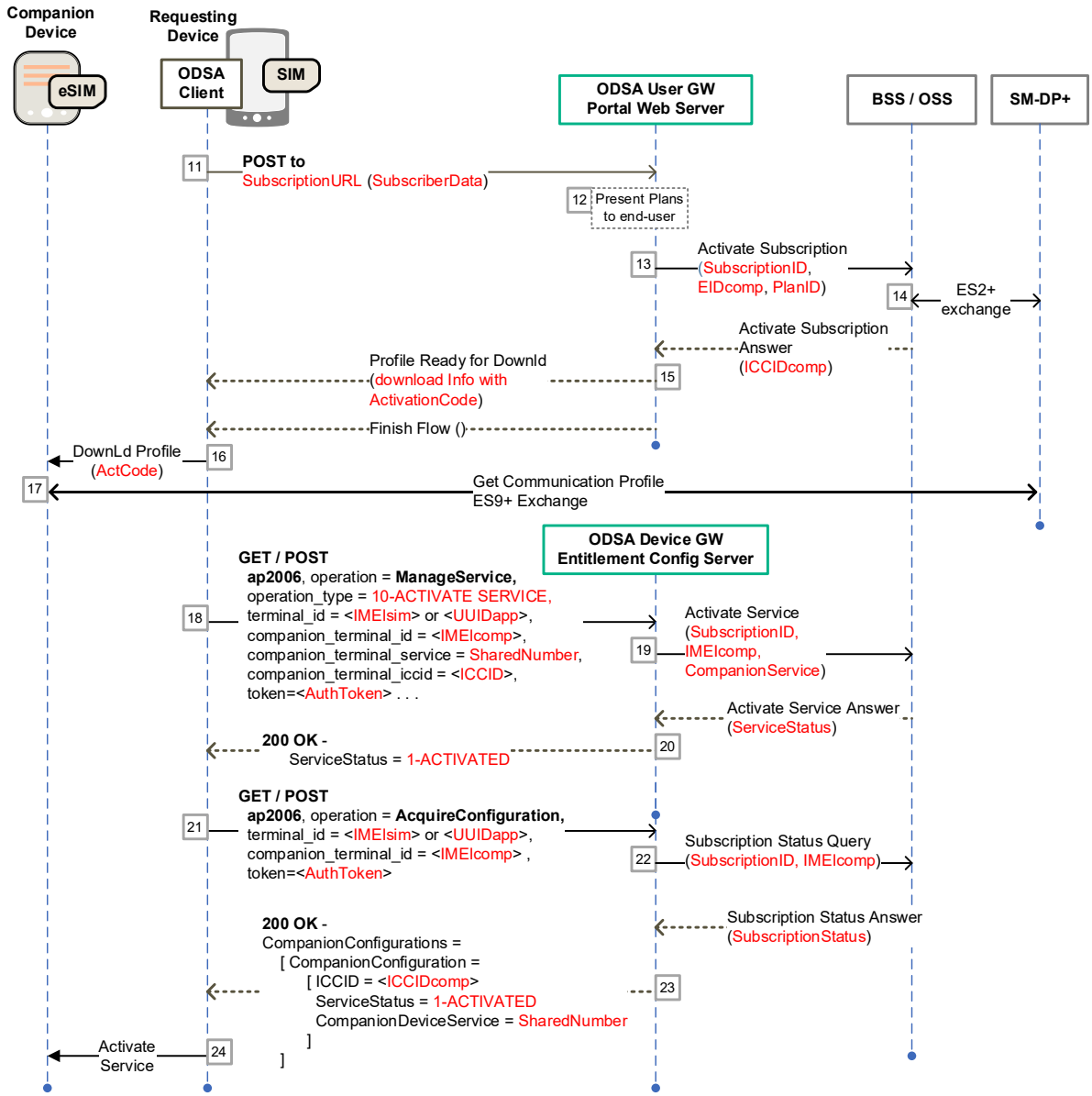

Figure 22. Final steps for companion ODSA procedure with profile download info from ODSA portal.

7.3 ODSA Portal with Delayed Download Info – Final Steps

The following presents the case where:

- The companion ODSA client application was already informed to use the SP's ODSA portal to complete the subscription procedure (refer to 7.1).
- The ODSA portal interacts with the end-user for plan selection and subscription activation but does not return the eSIM profile download information to the application.
- The companion ODSA client application subsequently obtains the eSIM profile information and service activation status by querying the ECS.
- The **ManageService** operation is not used by the companion ODSA application.

The finalization of the process depends on the usage of network-generated notification messages (refer to 2.6). If notifications are used, the device will continue with the push-

enabled procedure 7.3.1 . If notifications are not used, the ODSA client will go into polling (7.3.2).

7.3.1 ODSA Portal with Delayed Download Info – Final Steps - Push

The following presents the case where:

- The application registered for network-based event notification (refer to 2.6)

In this example the companion ODSA client has registered for push-notifications, so the application waits for a notification from the ECS when the eSIM profile is ready (refer to 2.6).

Figure 23 shows the final steps of the Companion ODSA procedure in the case where the eSIM profile download information is obtained by the application after the end-user interactions with the ODSA portal (delayed delivery). The steps are:

11. The ODSA client application connects with the ODSA portal web server using the URL provided in the **ManageSubscription** operation, allowing the web pages from the portal to be displayed to the end-user.
12. The ODSA portal web server presents a set of plan offers to the end-user and captures the selection from the end-user.
13. The ODSA portal makes a request towards the SP's back-end system to activate the selected plan and subscription.
14. The SP's back-end system interacts with the SM-DP+ over the ES2+ interface to make the required eSIM profile requests associated with the new subscription (for example, `DownloadOrder`, `ConfirmOrder` and `ReleaseProfile`), and indicates to the ODSA portal that the final response with the download info is delayed (asynchronous)
15. The ODSA portal indicates to the ODSA client application the end of the end-user flow via a JavaScript callback function without providing the eSIM profile download information (activation code)
16. The ODSA client application makes an **AcquireConfiguration** request to the ECS to verify that the subscription and service for the companion device are in the proper states. It also adds the `notif_token` and `notif_action` to the request, so that infrastructure-based notifications can be used.
17. The ECS queries the SP's back-end system managing the subscriptions and profiles.
    If the subscription is not yet ready and eSIM profile info is not yet available, go to step 18.
    If the subscription is ready, as well as eSIM profile download info, go to step 20
18. The ECS generates a 200 OK response with a `CompanionDeviceConfiguration` entry bearing the ACTIVATING status (value of 2). It also uses the `RegisterNotifStatus` parameter to notify the device about the Notification Registration (0 = SUCCESS).
19. The ODSA application now waits until it receives a new status by the ECS via the established notification mode.
20. After a delay, as soon as the ECS gets notified about a status change from the MNO-backend, the ECS notifies the ODSA application about a Status Change, using the

method defined in `notif_action`. The ODSA application therefore repeats the **AcquireConfiguration,** going to step 16

21. The ECS generates a 200 OK response with a `CompanionDeviceConfiguration` entry for the newly active subscription bearing the ACTIVATED status (value of 1) and a filled in `DownloadInfo` structure.
22. As the companion device's subscription and service are in the right state, the ODSA client application informs the companion device to download the eSIM profile.
23. The companion device downloads the eSIM profile from the SM-DP+
24. The ODSA client application informs the companion device to initiate cellular service.

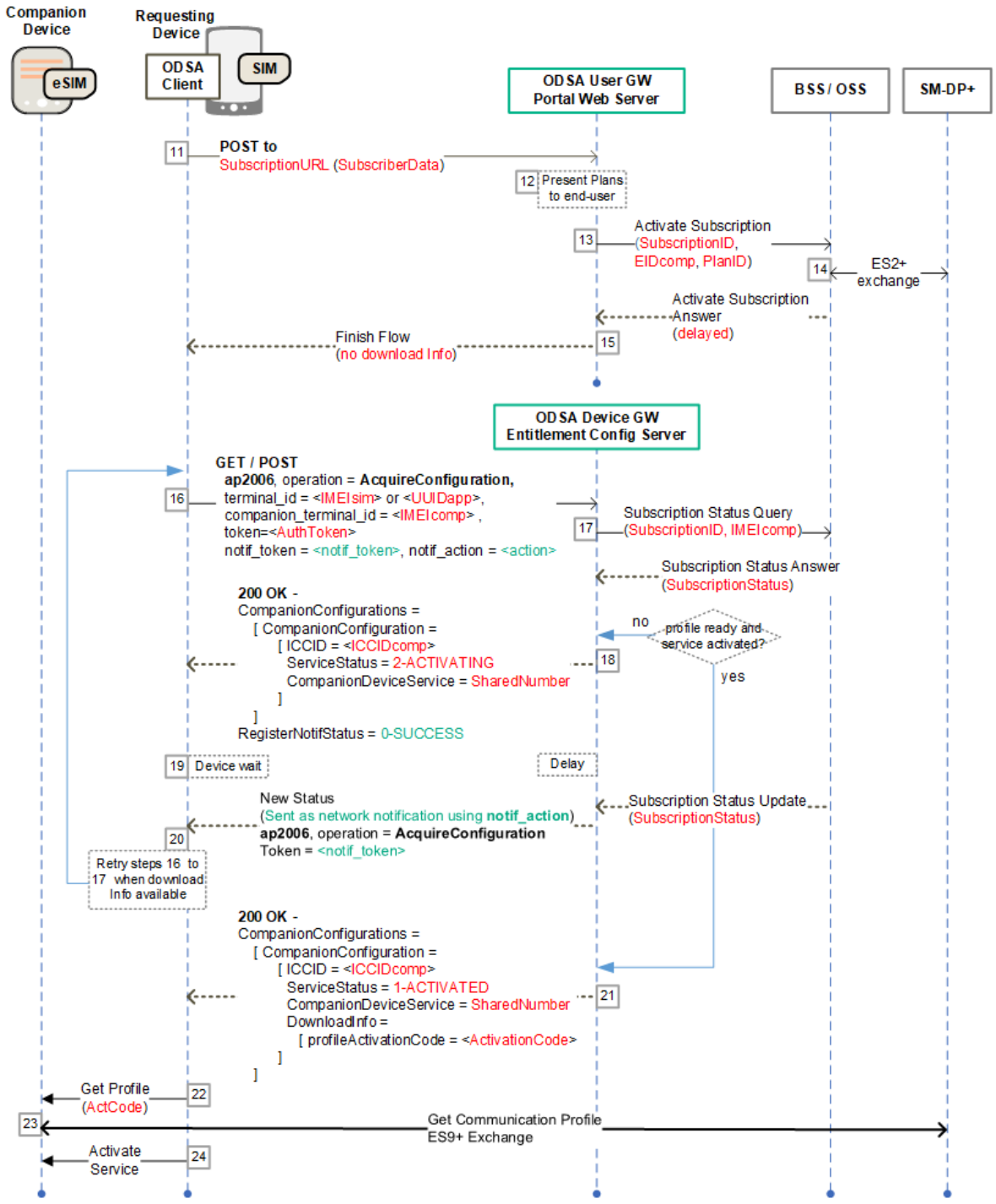

Figure 23. Final steps for companion ODSA procedure with ODSA portal, delayed profile download info and enabled push-notifications.

7.3.2 ODSA Portal with Delayed Download Info – Final Steps - Polling

The following presents the case where:

- The application did not register for event notification (refer to 2.6)

In this example the companion ODSA client application polls the ECS continuously until the profile is ready and the service status is correct. Alternatively, the application can register for ODSA events and wait for the network-generated notification message (refer to 2.6).

Figure 24 shows the final steps of the Companion ODSA procedure in the case where the eSIM profile download information is obtained by the application after the end-user interactions with the ODSA portal (delayed delivery). The steps are:

11. The ODSA client application connects with the ODSA portal web server using the URL provided in the **ManageSubscription** operation, allowing the web pages from the portal to be displayed to the end-user.
12. The ODSA portal web server presents a set of plan offers to the end-user and captures the selection from the end-user.
13. The ODSA portal makes a request towards the SP's back-end system to activate the selected plan and subscription.
14. The SP's back-end system interacts with the SM-DP+ over the ES2+ interface to make the required eSIM profile requests associated with the new subscription (for example, `DownloadOrder`, `ConfirmOrder` and `ReleaseProfile`), and indicates to the ODSA portal that the final response with the download info is delayed (asynchronous)
15. The ODSA portal indicates to the ODSA client application the end of the end-user flow via a JavaScript callback function without providing the eSIM profile download information (activation code)
16. The ODSA client application makes an **AcquireConfiguration** request to the ECS to verify that the subscription and service for the companion device are in the proper states.
17. The ECS queries the SP's back-end system managing the subscriptions and profiles.
    If the subscription is not yet ready and eSIM profile info is not yet available, go to step 18.
    If the subscription is ready, as well as eSIM profile download info, go to step 20
18. The ECS generates a 200 OK response with a `CompanionDeviceConfiguration` that could be different based on the number of refresh requests the device has made.

    a) #Request < `MaxRefreshRequest`. The `CompanionDeviceConfiguration` response will bear the ACTIVATING status (value of 2). A specific polling interval value could be sent to define the new polling interval of the device application to refresh the service status. This new polling interval value could be different to the previous one based on the number of requests made by the device.
    b) #Request = `MaxRefreshRequest`. The `CompanionDeviceConfiguration` response will bear the DEACTIVATED, NO REUSE status (value of 4). At this point, the activation flow is finished.

19. After a delay (polling interval defined in the previous response), the ODSA application repeats the **AcquireConfiguration,** going to step 16
20. The ECS generates a 200 OK response with a `CompanionDeviceConfiguration` entry for the newly active subscription bearing the ACTIVATED status (value of 1) and a filled in `DownloadInfo` structure.

21. As the companion device’s subscription and service are in the right state, the ODSA client application informs the companion device to download the eSIM profile.
22. The companion device downloads the eSIM profile from the SM-DP+
23. The ODSA client application informs the companion device to initiate cellular service.

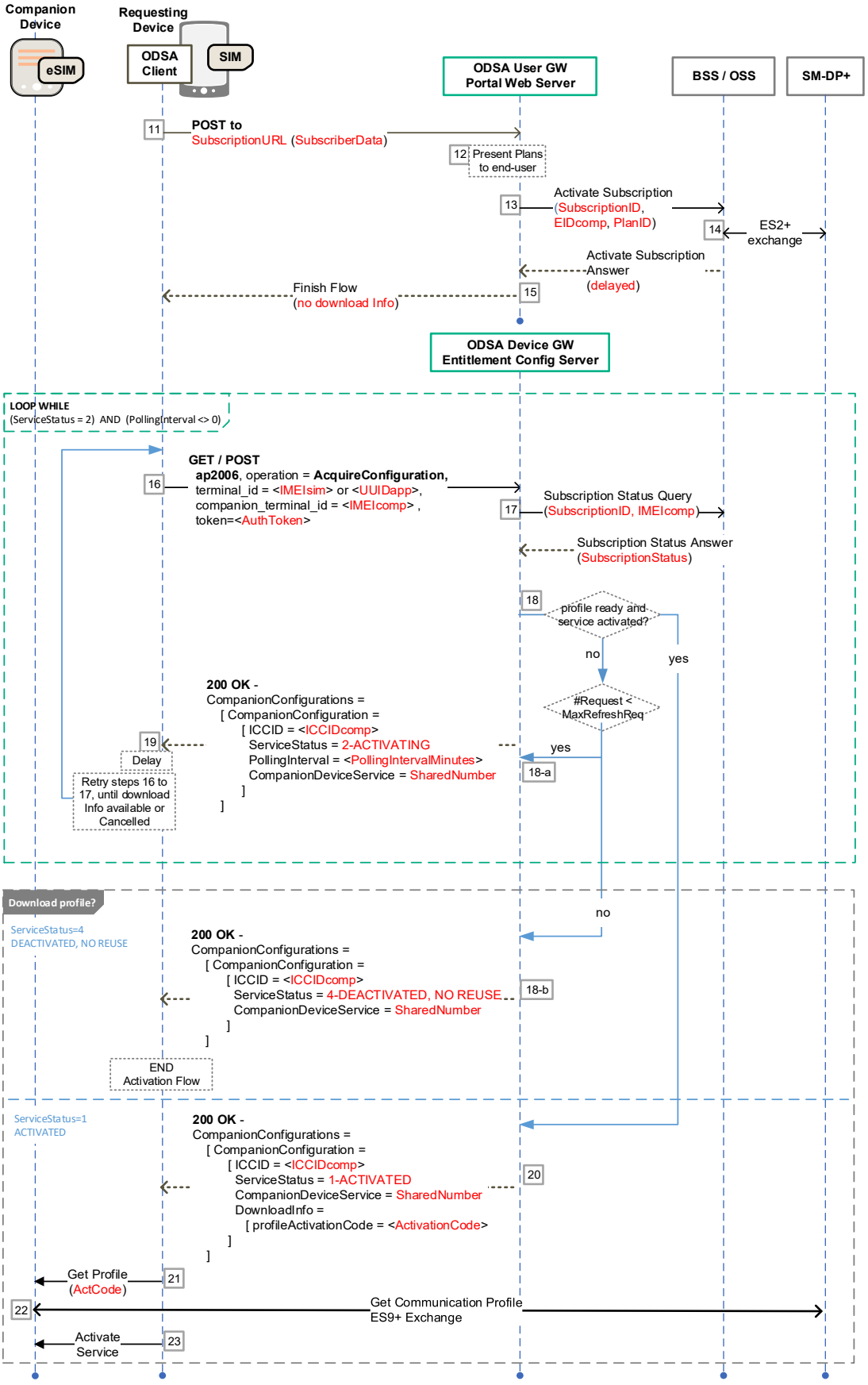

Figure 24. Final steps for companion ODSA procedure with ODSA portal and delayed profile download info.

7.4 Subscription Activation without ODSA Portal

The following presents the case where:

- The companion ODSA client application is allowed for the type of primary device and enabled for the end-user (entitled).
- The companion device does not have an active eSIM subscription/profile from the Service Provider.
- The SP is able to activate a subscription and create an eSIM profile for the companion device without involving the ODSA portal web server.

Figure 25 presents a call flow where the eSIM profile download information for the companion device is made available by the SP at the time of the **ManageSubscription** request. There is no need to send the end-user to an ODSA portal web server.

The steps 1 to 8 are the same as in 7.1. The remaining steps are:

9. The ECS queries the SP's back-end system to determine the next step and method to use for the companion device's subscription request (no need for ODSA portal)
10. The SP's back-end system interacts with the SM-DP+ over the ES2+ interface to make the required eSIM profile requests associated with the new subscription (for example, `DownloadOrder`, `ConfirmOrder` and `ReleaseProfile`) resulting in an activation code and ICCID for the companion device returned to the ECS.
11. The ECS processes the response from the SP's back-end system and generates the proper **ManageSubscription** 200 OK response with a `SubscriptionResult` set to DOWNLOAD_PROFILE (value of 2), and a filled in `DownloadInfo` structure.
12. The ODSA client application informs the companion device to download the eSIM profile.
13. The companion device downloads the eSIM profile from the SM-DP+
14. The ODSA client application makes an **AcquireConfiguration** request to the ECS to verify that the subscription and service for the companion device are in the proper states.
15. The ECS queries the SP's back-end system managing the subscriptions and profiles.
16. The ECS processes the response from the SP's back-end system and generates the proper 200 OK response containing `CompanionDeviceConfigurations` with a `CompanionDeviceConfiguration` entry for the newly active subscription bearing the ACTIVATED status (value of 1).
17. The ODSA client application informs the companion device to initiate cellular service.

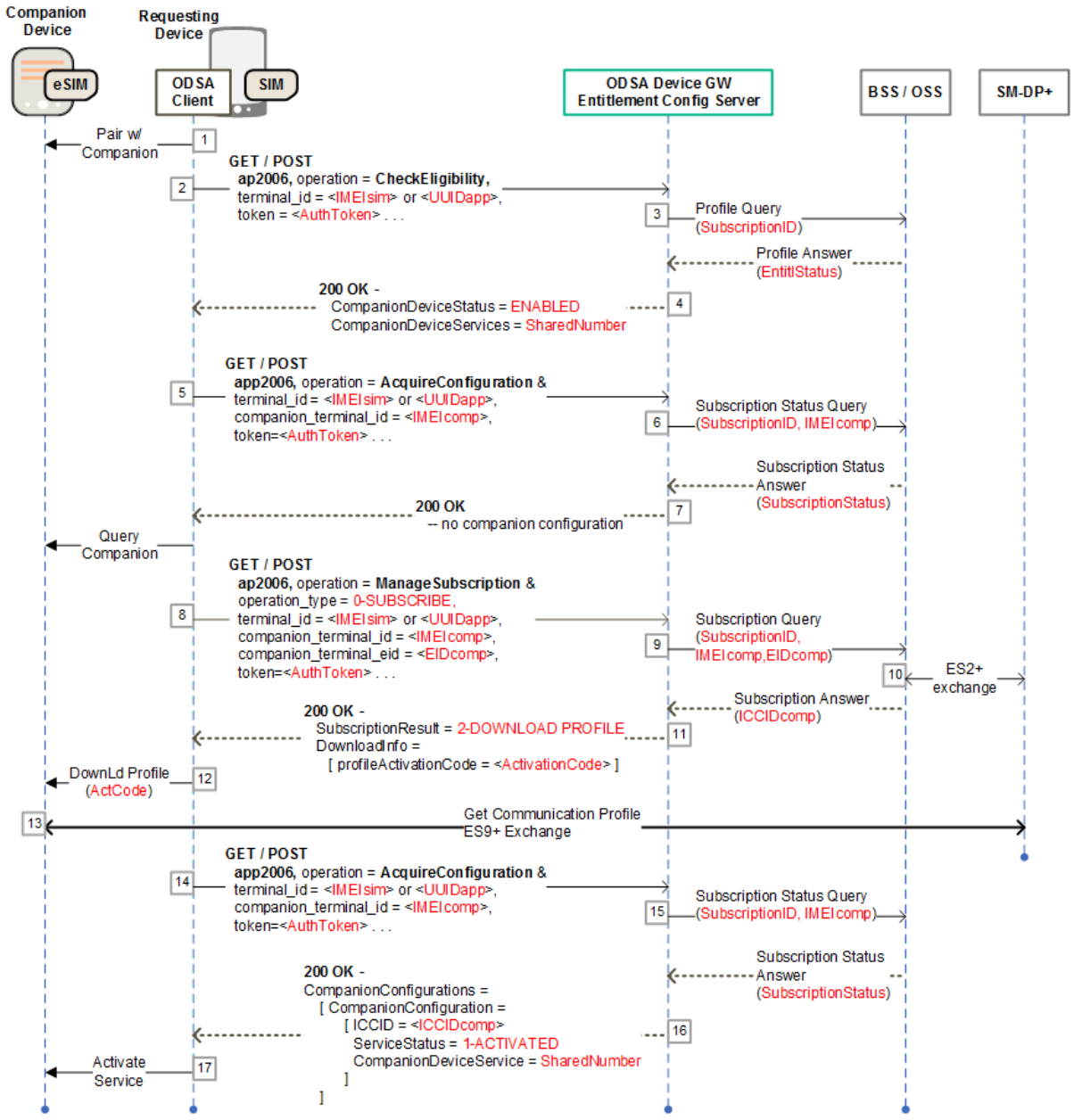

Figure 25. Call flow for Companion ODSA procedure without ODSA Portal

7.5 Subscription Pre-activation via another Channel

The following presents the case where:

- The companion ODSA application is allowed for the type of primary device and enabled for the end-user (entitled).
- The companion device has an active eSIM subscription and communication profile from the Service Provider, created beforehand through another channel (for example point of sale or call to a SP's representative).

Figure 26 presents a call flow where the eSIM profile download information for the companion device is made available by the SP at the time of the AcquireConfiguration request. There is no need to send the end-user to an ODSA portal web server.

The steps 1 to 4 are the same as in 7.1. The remaining steps are:

5. The ODSA client application makes an **AcquireConfiguration** request to the ECS to verify that the subscription and service for the companion device are in the proper states.
6. The ECS queries the SP's back-end system managing the subscriptions and profiles, which shows that the companion device already has a subscription and associated eSIM profile.
7. The ECS processes the response from the SP's back-end system and generates the proper 200 OK response containing `CompanionDeviceConfigurations` with a `CompanionDeviceConfiguration` entry for the newly active subscription bearing the ACTIVATED status (value of 1) and a filled in `DownloadInfo` structure.
8. The ODSA client application informs the companion device to download the eSIM profile.
9. The companion device downloads the eSIM profile from the SM-DP+
10. The ODSA application informs the companion device to initiate cellular service.

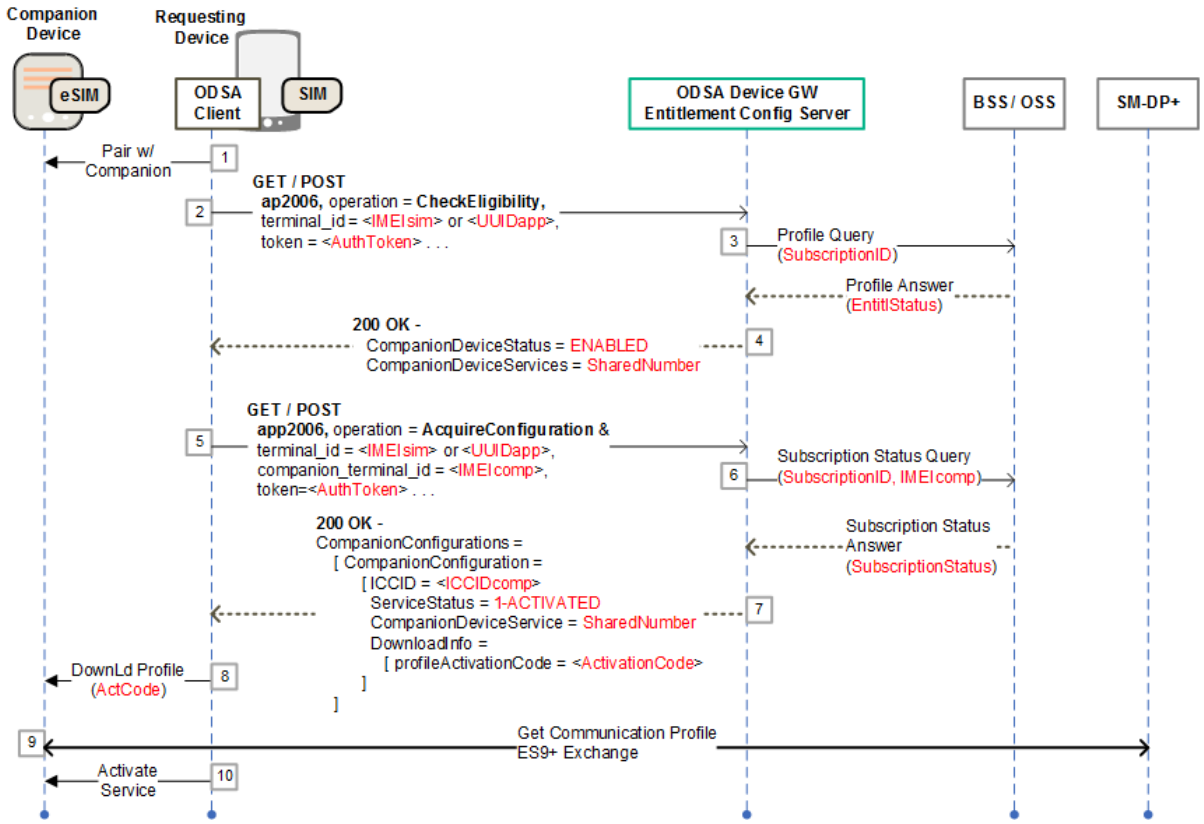

Figure 26. Call flow for Companion ODSA procedure with pre-activated subscription.

7.6 Multiple companion device Management without webview

The following presents the case where:

- The Companion ODSA device is entitled, and companion devices are entitled by the Service provider for service.
- The end-user requests the current eSIM profiles associated with the active subscription and their installation location information and there is no need to involve the SP's ODSA portal web server.
- The end-user requests subscription transfer on the old companion device which carries an active subscription with the SP.

Figure 27 shows the steps of the flow for the transfer of an active companion eSIM profile from one companion to another. The Companion ODSA app acquires proper entitlement and subscription data from the SP's ECS. The steps are:

1. User requests On-Device Activation via the companion client application and obtains the necessary companion device information.
2. The Companion ODSA client application makes a **CheckEligibility** request to the ECS.
3. The ECS queries the SP back-end system managing the entitlements and eSIM profiles associated with ODSA applications.
4. The ECS generates proper response with application status (ENABLED)
5. The Companion ODSA client application sends an **AcquireConfiguration** request to the ECS to query all of the eSIM profiles associated with the SubscriptionID.
6. (OPTIONAL) The ECS queries the BSS to get the `SubscriptionStatus` of all the eSIM profiles associated with the SubscriptionID.
7. The ECS confirms that the `terminal_id` doesn't support websheet and therefore lists all `companion_terminal_iccid` values and their associated device information in the **CompanionConfigurations** parameter along with the associated CompanionDeviceInfo element.
8. The Companion ODSA client may list the eSIM profiles and their installation location to the User and may provide an MMI to allow the user to manage the location of these eSIM profiles by ICCID. The user selects the ICCID they want transferred and sets the **`old_companion_terminal_id`** and **`old_companion_terminal_ICCID`** using this MMI.
9. The Companion ODSA client application sends a **ManageSubscription** request to the ECS to start the subscription procedure with the SP. If `old_companion_terminal_iccid` is not present the ECS recognizes that the Companion ODSA client application doesn't support a device management MMI, the ES shall redirect the Companion ODSA client to the websheet as described in clause 6.5.3. If the ECS doesn't support any form of eSIM management function and `old_companion_terminal_iccid` is present the ECS shall follow the error mechanisms defined in section 2.10 of this document.
10. The ECS requests for a new subscription from the SP's back-end system to complete the transfer.
11. A set of eSIM profile requests over the ES2+ interface (for example, `DownloadOrder`, `ConfirmOrder` and `ReleaseProfile`) is made to the SM-DP+,

resulting in an activation code and ICCID (ICCIDnew) of the eSIM profile to be downloaded onto the new companion device.

12. The ECS sends subscription information (details of the eSIM profile) back to the Companion ODSA client along with subscription result (done).
13. The Companion ODSA client sends the activation code to the new Companion device to being the eSIM profile download.

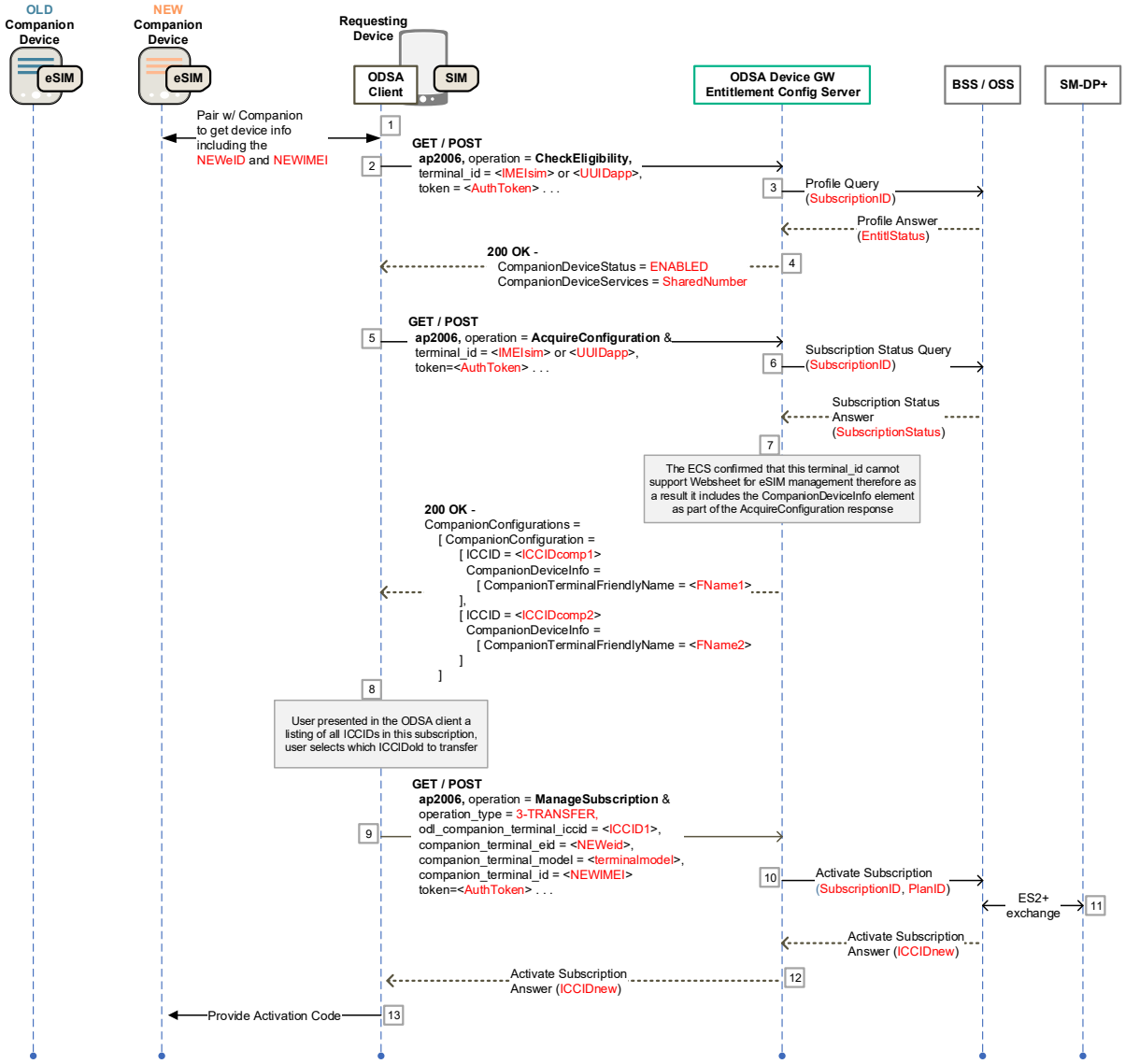

Figure 27: Companion ODSA procedure for multiple companion device Management without webview (eSIM transfer to new device)

7.7 Early eligibility check with OIDC and web portal

The following presents the case where:

- The companion ODSA application is allowed for the type of requesting device.
- The companion device has not yet an active eSIM subscription/ profile from the Service Provider.
- The companion device model is **not supported** by the Service Provider.

- The authentication mechanism is based on OAuth 2.0 / OpenID Connect.
- The SP's ODSA portal web server is responsible for completing the subscription activation for the companion device.

Figure 28 presents a call flow where the user is advised about the incompatibility of the model without needing to authenticate, with a 302 HTTP redirect mechanism. These steps are:

1. End-user invokes the Companion ODSA client application on the requesting device which connects with the companion device to initiate the ODSA procedure (over a protocol outside the scope of this specification).
2. The initial GET request described in 2.8.2 includes the `companion_terminal_id` parameter, set to the companion device IMEI. There is no token as it is the first time the user is trying to entitle the companion device.
3. The ECS determines that the companion device is not eligible to the service and returns the HTTP 302 redirect answer to indicate the “not enable” web page to the ODSA client.
4. and 5. this later can display a web page explaining the issue to the end-user. The page may be closed with a call to the dismissFlow() callback.

The benefit of this use case is to keep the user journey simple by checking first the device compatibility before asking the user to authenticate. Nevertheless, this requires the client to provide the optional `companion_terminal_id` parameter, initialized with its IMEI. When this optional use case is implemented, devices not providing the `companion_terminal_id` parameter are still managed as described in Figure 3.

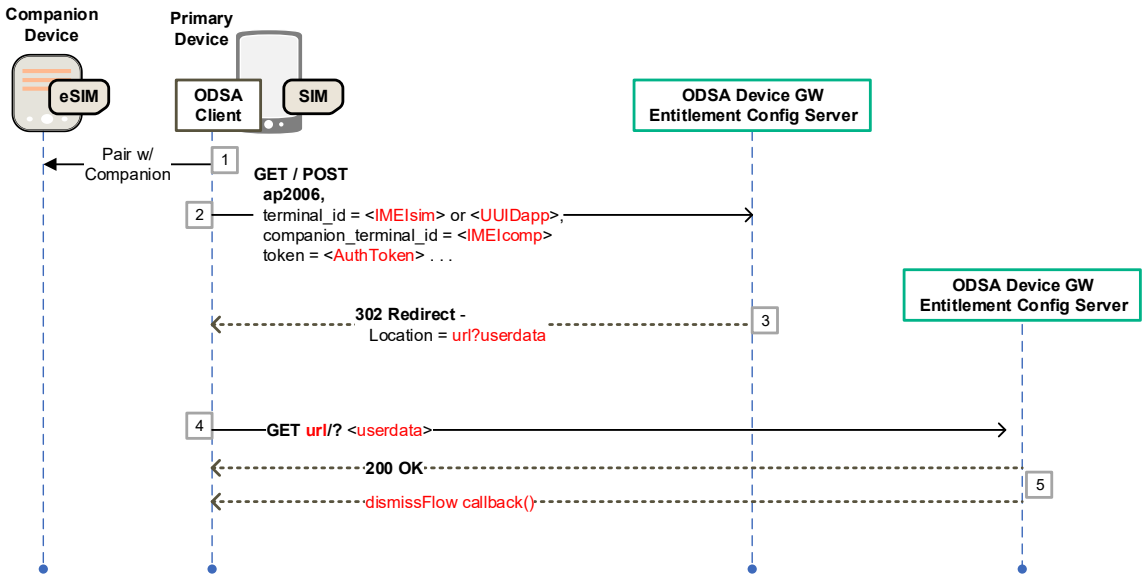

Figure 28. Companion device incompatibility detected by companion_terminal_id.

8 Primary ODSA Procedure Call Flows

The following sections present a number of informational call flows for the different user experiences and use cases of the Primary ODSA procedure. The ODSA application on the primary device is invoked at the request of the end-user and should capture proper user consent in order to have access to the eSIM on that primary device.

The exchanges between the Entitlement Configuration Server (ECS) (aka ODSA Device Gateway) and the Service Provider's (SP) back-end systems are shown for informational purposes only. This applies as well for the exchanges that involve the ODSA Portal Web Server.

8.1 New eSIM Subscription Activation via ODSA Portal

The following presents the case where:

- The Primary ODSA client application is allowed for the type of primary device and enabled by the SP (entitled).
- The primary device does not have an active eSIM subscription/profile from the SP and the end-user does not have a subscription on another device.
- The SP supports the OpenID Connect authentication flow, which also includes a "create account" option for new subscription request.
- The SP's ODSA portal web server is responsible for completing the subscription activation for the primary device's eSIM.

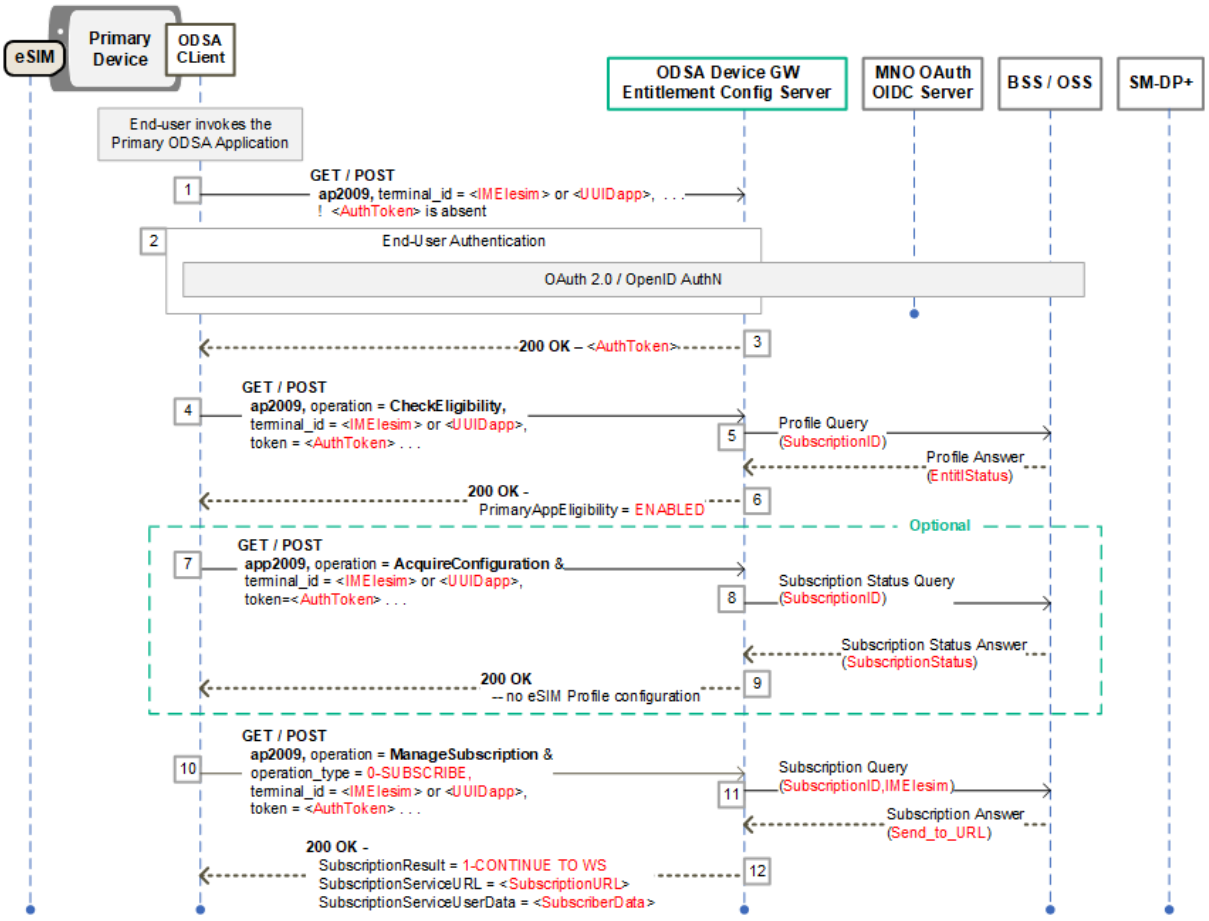

Figure 29 shows the initial steps of the flow for the activation of a new subscription leveraging the SP's ODSA portal. The Primary ODSA client application acquires proper entitlement and subscription data from the SP's ECS. The steps are:

1. User requests On-Device Activation via the Primary ODSA client application. This request may be done by clicking URI or scanning the QR code format, e.g., on the SP's web portal, as defined in section 6.9.2. The client sends an initial POST or GET request with proper terminal parameters to the ECS.
2. As there is no parameter associated with authentication or identification, the ECS invokes OAuth/OpenID authentication and connects the app/end-user with the SP's OpenID/OAuth 2.0 platform.
3. At the conclusion of the Authentication (which includes account creation steps), the ECS receives proper ID and access tokens from the OpenID platform and returns an ECS-generated AuthN Token to the ODSA application (see 2.8.2 for details)
4. The Primary ODSA client application makes a **CheckEligibility** request to the ECS.
5. The ECS queries the SP back-end system managing the entitlements and profile associated with ODSA applications.
6. The ECS generates proper response with application status (ENABLED)
7. Optional - Since the target service is allowed, the Primary ODSA application sends an **AcquireConfiguration** request to the ECS to obtain information on any eSIM profiles associated with the device.
8. The ECS queries the SP's back-end system managing the subscriptions and active profiles.
9. The ECS processes the response from the SP's back-end system and generates the proper 200 OK response without any `PrimaryDeviceConfigurations` (no eSIM profile/subscription is associated with the device).
10. The Primary ODSA client application sends a **ManageSubscription** request to the ECS to start the subscription procedure with the SP.
11. The ECS queries the SP back-end system responsible for managing subscriptions and makes a request for a new subscription.
12. The ECS generates a proper response with the subscription procedure data. It contains a `SubscriptionResult` set to CONTINUE_TO_WS (value of 1), and `SubscriptionServiceURL` along with `SubscriptionServiceUserData` presenting the URL of the ODSA Portal web server and any user-specific data that would be useful to the Portal.

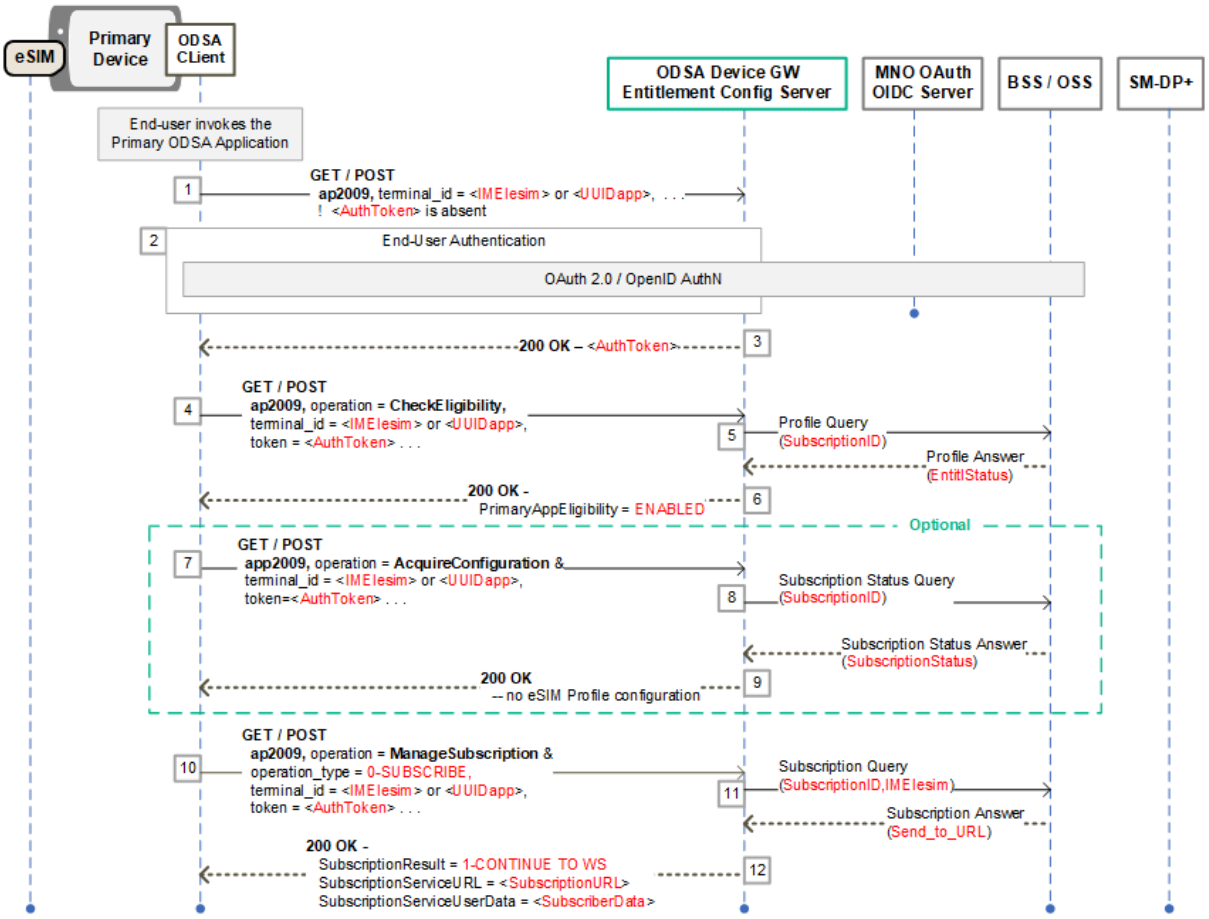

Figure 29. Primary ODSA procedure for New Subscription involving ODSA Portal – Initial Steps

Figure 30 presents the final steps of the flow for the activation of a new subscription leveraging the SP's ODSA portal. The Primary ODSA app connects the end-user to the SP's ODSA Portal to finalize the subscription activation. The steps are:

13. The Primary ODSA device application sends the end-user to the SP's ODSA web server portal.
14. The SP ODSA portal captures the subscription and plan selection from the end-user.
15. The SP's back-end system managing subscription receives a new subscription request from the SP portal.
16. A set of eSIM profile requests over the ES2+ interface (for example, `DownloadOrder`, `ConfirmOrder` and `ReleaseProfile`) is made to the SM-DP+, for the new subscription associated with the device eSIM, resulting in an activation code and ICCID for the primary device.
17. Via a JavaScript call back function, the SP ODSA portal sends subscription information (details of the eSIM profile) back to the Primary ODSA app.
18. The Primary ODSA device application informs the eSIM to download the eSIM profile, which is obtained from the SM-DP+.
19. The device's eSIM gets the eSIM profile from the SM-DP+ via ES9+ channel.

20. Optional - The Primary ODSA app makes another **ManageSubscription** to the ECS to provide/confirm the download of the newly created ICCID and to validate that the primary device subscription is ready and in proper activated state.
21. The ECS queries the Subscription Management system.
22. The ECS generates the proper response with subscription result (3-DONE).
23. Optional - The Primary ODSA client application makes an **AcquireConfiguration** request to the ECS to verify that the subscription and service for the device are in the proper states.
24. The ECS queries the SP's back-end system managing the subscriptions and profiles.
25. The ECS processes the response from the SP's back-end system and generates the proper 200 OK response containing a `PrimaryConfiguration` entry for the newly active subscription bearing the ACTIVATED status (value of 1).
26. As the primary device's subscription and service is in the right state, the primary device can initiate cellular service.

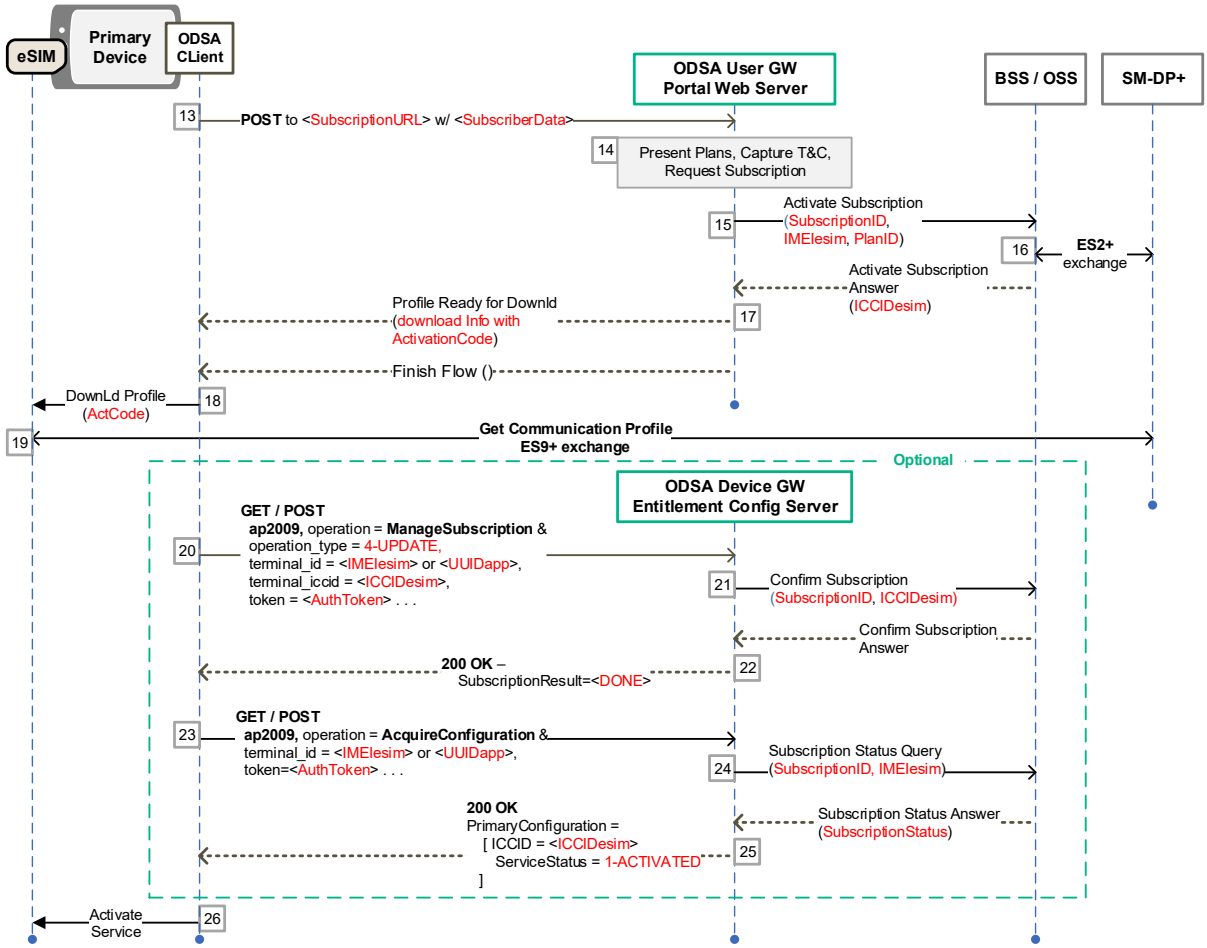

Figure 30. Primary ODSA procedure for New Subscription involving ODSA Portal – Final Steps

8.2 Additional eSIM Subscription Activation via ODSA Portal

The following presents the case where:

- The Primary ODSA device application is allowed for the type of primary device and enabled by the SP (entitled).

- The primary device already carries an active subscription and communication profile from the SP, accessible on a SIM.
- The SP's ODSA portal web server is responsible for completing the subscription activation for the primary device's eSIM.

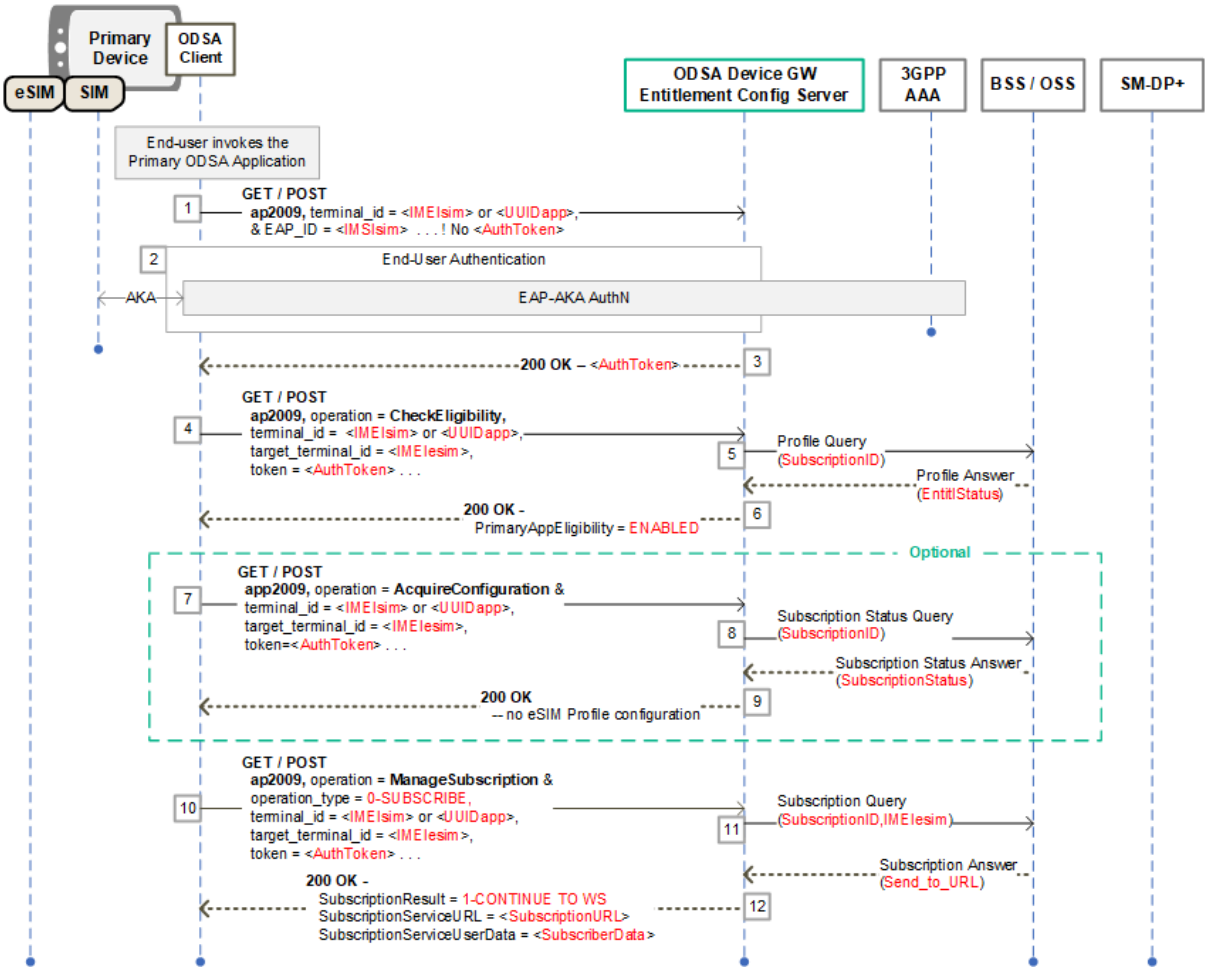

Figure 31 shows the initial steps of the flow for the activation of an additional subscription leveraging the SP's ODSA portal. The Primary ODSA device application acquires proper entitlement and subscription data from the SP's ECS.

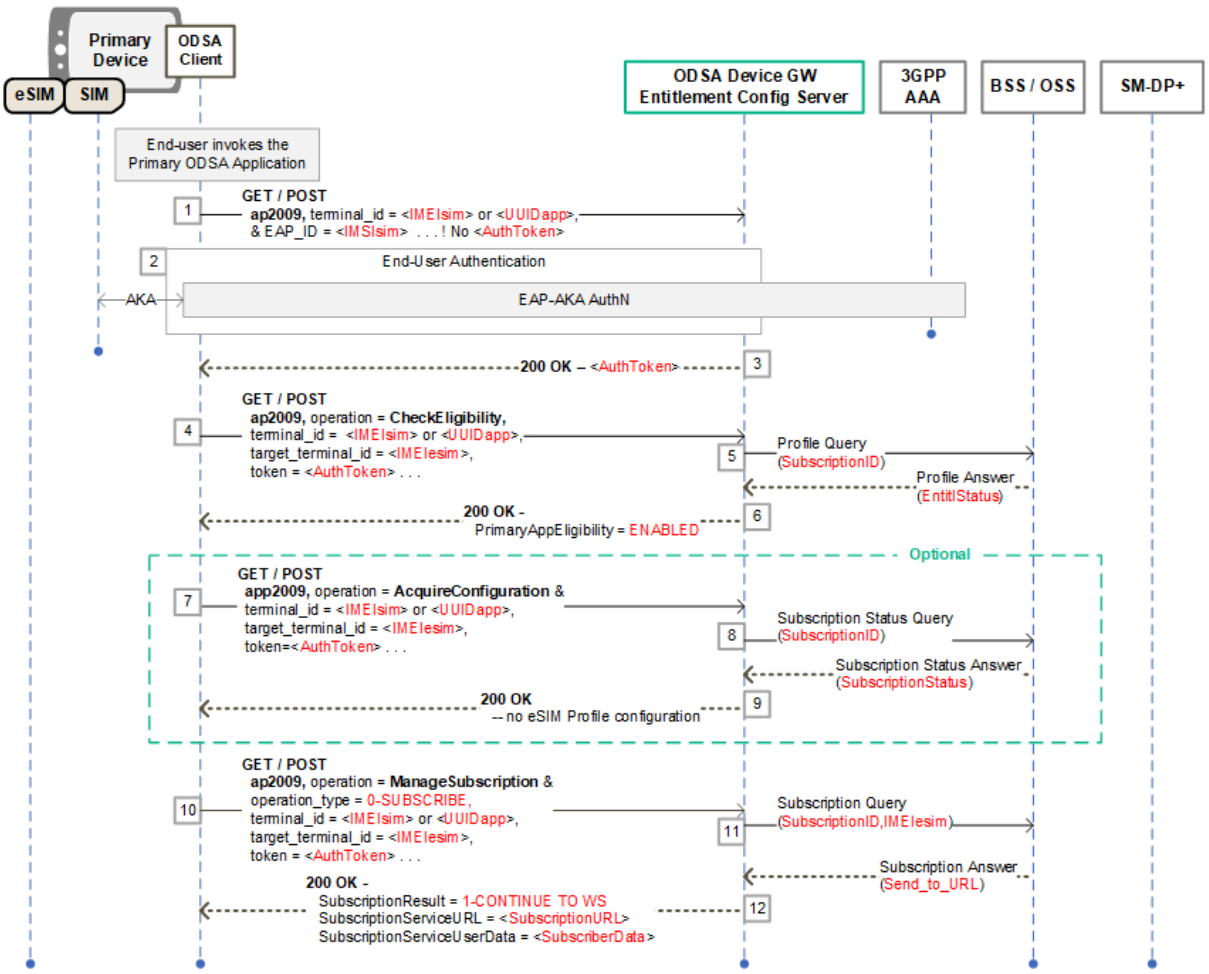

Figure 31. Primary ODSA procedure for Additional Subscription involving ODSA Portal – Initial Steps

Figure 32 shows the final steps of the flow where the Primary ODSA app connects the end-user to the SP's ODSA Portal to finalize the subscription activation.

The steps are:

1. User requests On-Device Activation via the Primary ODSA application that sends an initial POST or GET request with proper terminal parameters to the ECS. The request contains the EAP_ID parameter, indicating that the app has access to a SIM or eSIM with an active subscription/profile.
2. The ECS initiates the EAP-AKA authentication procedure and performs the proper EAP-AKA exchange with the application (see 2.8.1 for details)
3. At the conclusion of the Authentication the ECS returns an ECS-generated AuthN Token to the ODSA application
4. **Steps 4 to 26** are the same as in clause 8.1.
   The difference is the addition of the `target_terminal_id` parameter for **CheckEligibility**, **AcquireConfiguration** and **ManageSubscription**, carrying the device identifier for the eSIM. The `terminal_id` parameter carries the device identifier for the SIM with the active subscription.

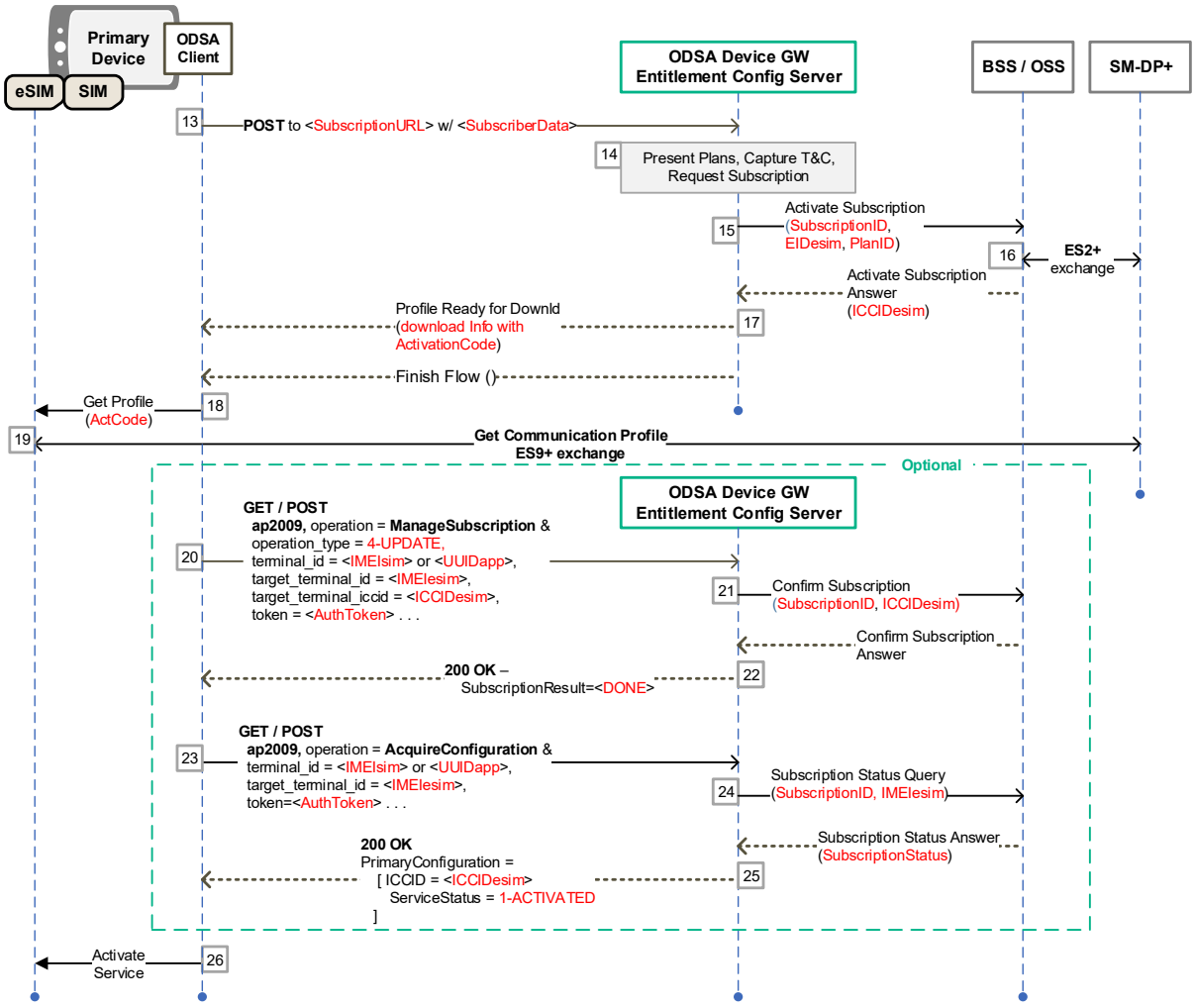

Figure 32. Primary ODSA procedure for Additional Subscription involving ODSA Portal – Final Steps

8.3 Subscription Transfer with OTP – initial steps

The following presents the case where:

- The Primary ODSA device application is allowed for the type of primary device and enabled by the SP (entitled).
- The end-user has an active subscription with the SP identified by its MSISDN.
- There is no need to involve the SP's ODSA portal web server as the same type of subscription and plan is activated on the new device.

Figure 33 shows the steps of the flow for the activation of a subscription based on an existing subscription validated with a One-Time Password (OTP). The steps are:

1. User requests On-Device Activation via the Primary ODSA client application. The client discovers that the end-user wants to transfer an existing subscription and sends an initial request to the ECS, which includes proper terminal and `msisdn` parameters.

2. The ECS performs OTP-based authentication by sending an OTP to the end-user (any method can be used, like SMS or e-mail) and returns a new `Cookie` to the client.
3. The Primary ODSA client application captures the OTP from the end-user and relays it to the ECS with another request, this time with `otp` parameter and proper `Cookie`.

The ECS validates the received OTP and generates response with new ECS-generated Authentication Token back to the client application.

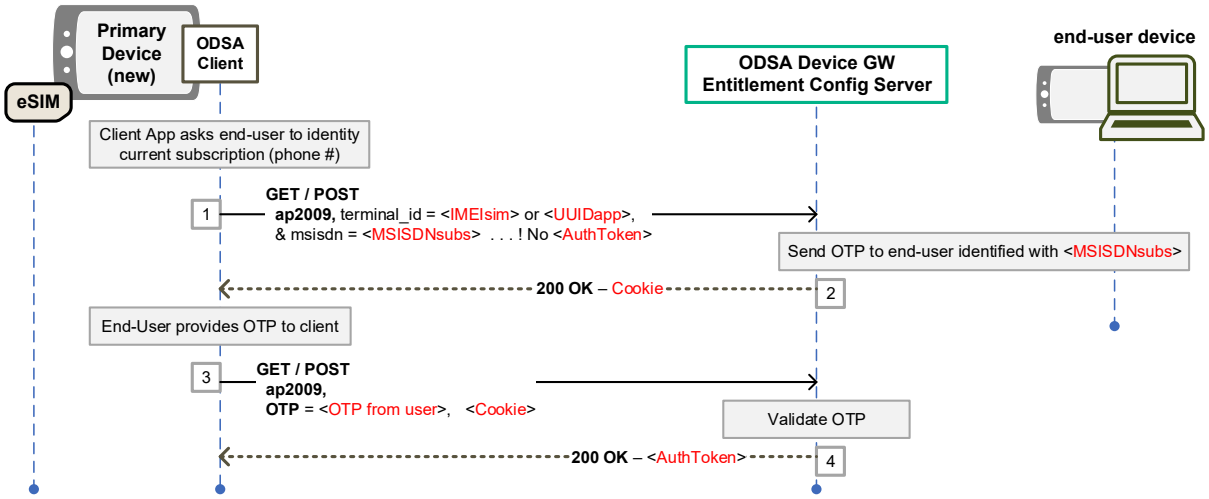

Figure 33. Primary ODSA procedure for Subscription Transfer with OTP

8.4 Subscription Transfer with OAuth/OpenID – initial steps

The following presents the case where:

- The Primary ODSA device application is allowed for the type of primary device and enabled by the SP (entitled).
- The end-user has an active subscription with the SP, but cannot receive an OTP via SMS due to, e.g., the end-user has their device (including eSIM and/or pSIM) lost and/or stolen.
- There is no need to involve the SP's ODSA portal web server as the same type of subscription and plan is activated on the new device.

Figure 34 shows the initial steps of the flow for the activation of a subscription based on an existing subscription validated via OAuth or OpenID. The steps are:

1. User requests On-Device Activation via the Primary ODSA client application that sends an initial POST or GET request with proper terminal parameters to the ECS.
2. As there is no parameter associated with authentication or identification, the ECS invokes OAuth/OpenID authentication by redirecting the flow to the SP's OAuth 2.0/OpenID platform (using a 302 Found/Redirect response)
3. Authentication of the end-user by the SP's OpenID/OAuth 2.0 platform is performed, using proper SP-selected authenticators (see 2.8.2 for details)

4. At the conclusion of the Authentication, the ECS receives proper ID and access tokens from the OpenID platform and returns an ECS-generated AuthN Token to the ODSA application.

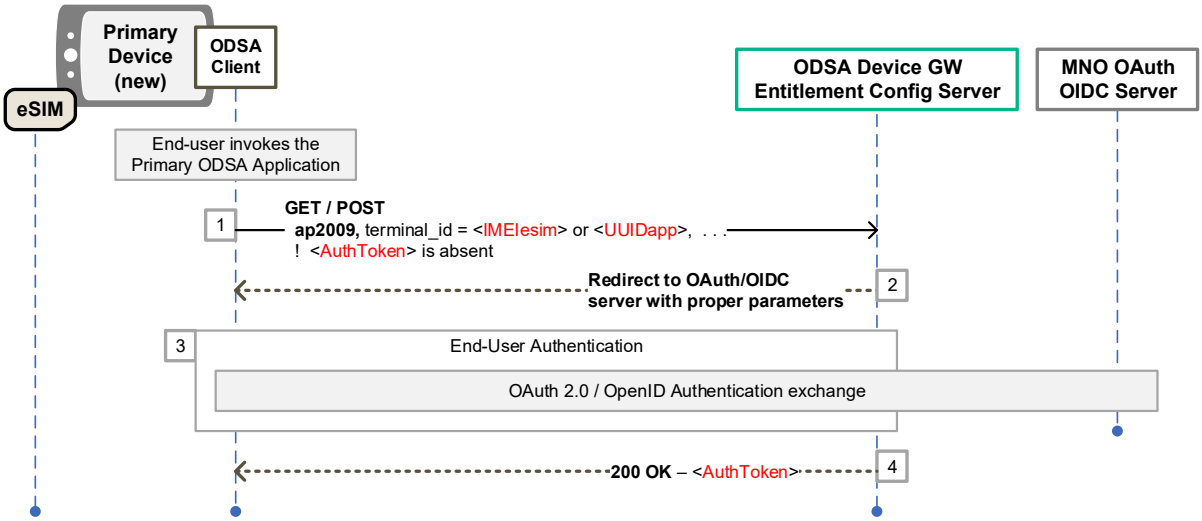

Figure 34. Primary ODSA procedure for Subscription Transfer with OAuth/OpenID

8.5 Subscription Transfer with OTP or OAuth/OpenID– final steps

The following presents the case where:

- The Primary ODSA device application is allowed for the type of primary device and enabled by the SP (entitled).
- The end user is already authenticated using a method described in 8.3 or 8.4;

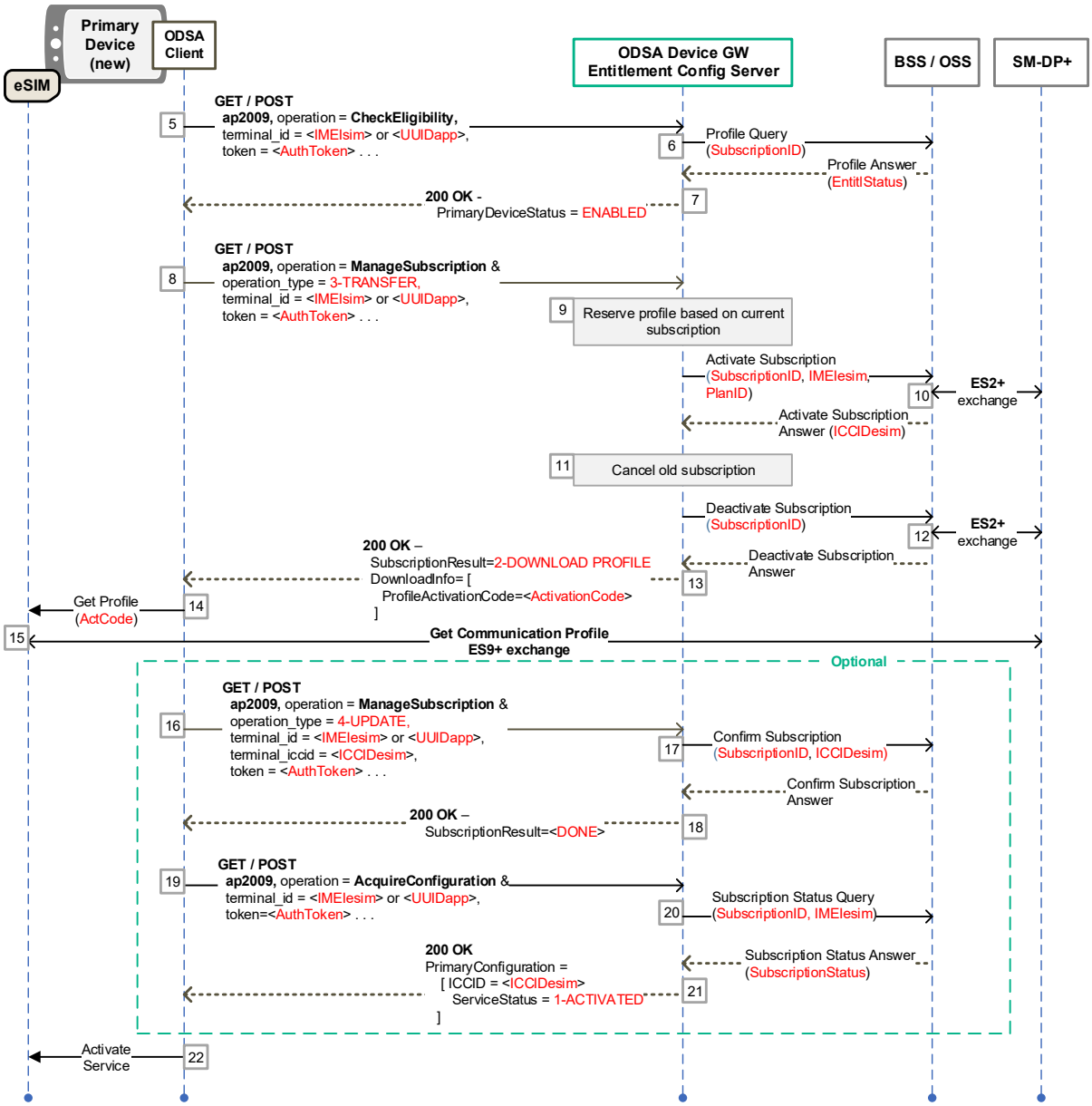

Figure 35 shows the steps of the flow for the activation of a subscription based on an existing subscription:

Steps 1 to 4 handle the authentication as shown in clause 8.3 or 8.4 .

5. The Primary ODSA client application makes a **CheckEligibility** request to the ECS.
6. The ECS queries the SP back-end system managing the entitlements and profile associated with ODSA applications.
7. The ECS generates proper response with application status (ENABLED)
8. The Primary ODSA client application sends a **ManageSubscription** request to the ECS to start the subscription procedure with the SP. If the `old_terminal_iccid` is available, the device should also add the parameter.
9. The ECS checks with the BSS/OSS to verify if there is more than one device/ICC linked with the subscription. If the `old_terminal_iccid` is available, the ECS

checks this value for correctness. If an identifier for the old terminal is needed, the ECS obtains it using e.g. the Websheet procedure in chapter 8.6.

10. The ECS requests for a new subscription from the SP's back-end system. A set of eSIM profile requests over the ES2+ interface (for example, `DownloadOrder`, `ConfirmOrder` and `ReleaseProfile`) is made to the SM-DP+, for the new subscription associated with the primary device eSIM, resulting in an activation code and ICCID for the primary device.
11. The ECS requests for a subscription cancellation from the SP's back-end system.
12. A set of eSIM profile requests over the ES2+ interface is made to the SM-DP+, to cancel the current subscription.
13. The ECS sends subscription information (details of the eSIM profile) back to the app along with subscription result (2-DOWNLOAD PROFILE).
14. The primary ODSA client application informs the eSIM to download the profile.
15. The device's eSIM gets the profile from the SM-DP+ via ES9+ channel.
16. Optional - The Primary ODSA client application makes another **ManageSubscription** to the ECS to provide/confirm the download of the newly created ICCID and to validate that the primary device subscription is ready and in proper activated state.
17. The ECS queries the Subscription Management system.
18. The ECS generates the proper response with subscription result (3-DONE).
19. Optional - The Primary ODSA client application makes an **AcquireConfiguration** request to the ECS to verify that the subscription and service for the new device are in the proper state.
20. The ECS queries the SP's back-end system managing the subscriptions and profiles.
21. The ECS processes the response from the SP's back-end system and generates the proper 200 OK response containing a `PrimaryConfiguration` entry for the newly active subscription bearing the ACTIVATED status (value of 1).
22. As the primary device’s subscription and service is in the right state, the primary device can initiate cellular service.

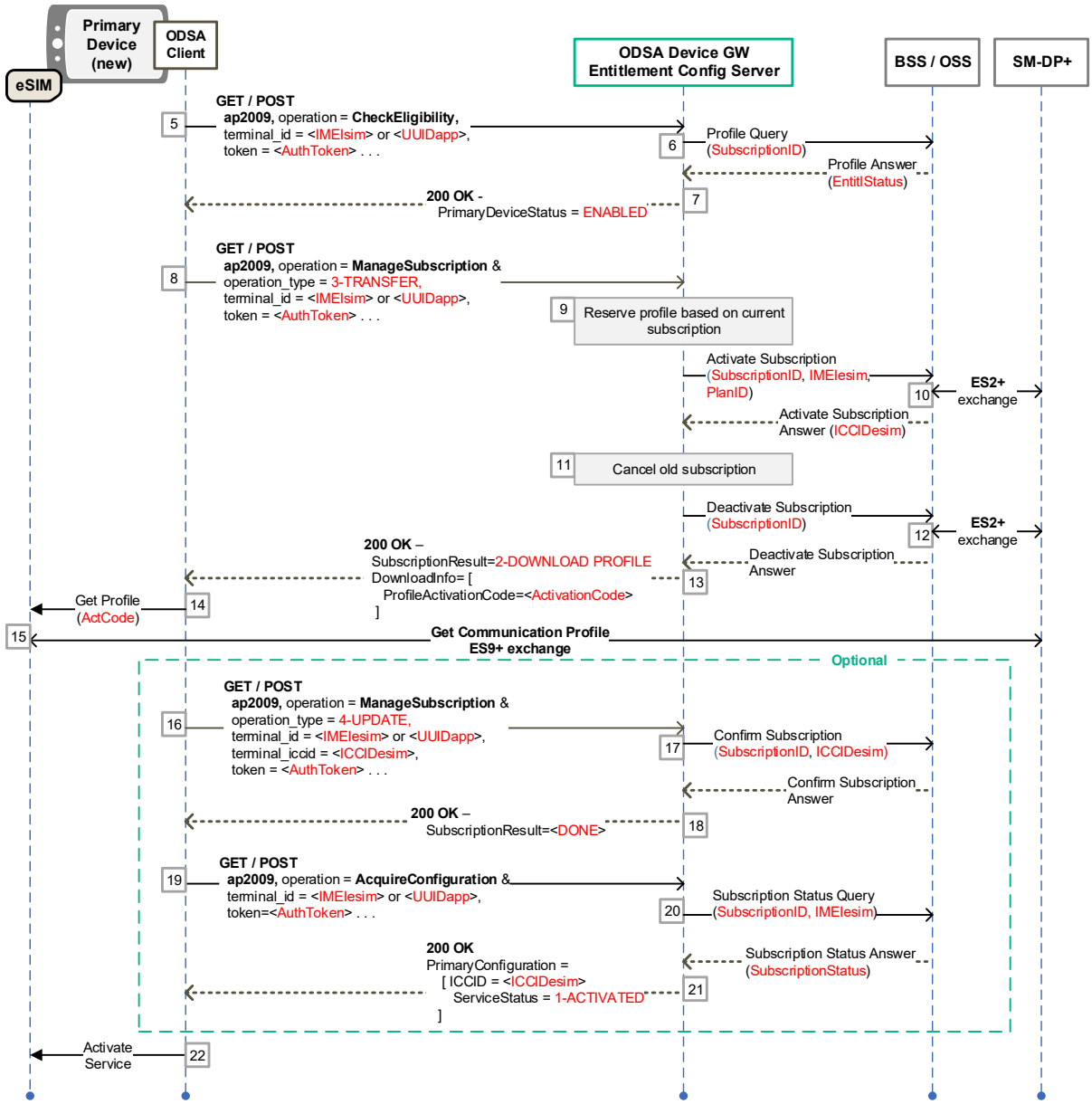

Figure 35. Primary ODSA procedure for Subscription Transfer with OAuth/OpenID

8.6 Using Websheet in eSIM Transfer

During the eSIM transfer process, and independently which device (old or new) triggers the request, it could be necessary to have some interaction with the user. In most cases, it could be done through a Websheet.

Figure 36 shows, as an example, the procedure where the old terminal identifier is needed in a subscription transfer procedure, and is obtained using a Websheet:

1. The Primary ODSA client application sends a request to the ECS to start the subscription procedure with the SP, in this case a **ManageSubscription**.
2. The ECS needs an identifier for the old device, as it was not included in the request. The ECS redirects the ODSA client to the WebServer.

3. The ODSA client requests the Web Server using the URL and UserData received in step 10.
4. The web server presents the user the active subscriptions, so that the user can choose the one he claims as his old device.
5. The Web Server uses the `SelectionCompleted` callback and returns the identifiers `old_terminal_id` and/or `old_terminal_iccid` to the ODSA client.
6. The Primary ODSA client application sends its initial request from step 1, adding the identifiers received in step 5.

Alternatively to step 5 and 6, the webserver can also forward the identifier to the ECS directly.

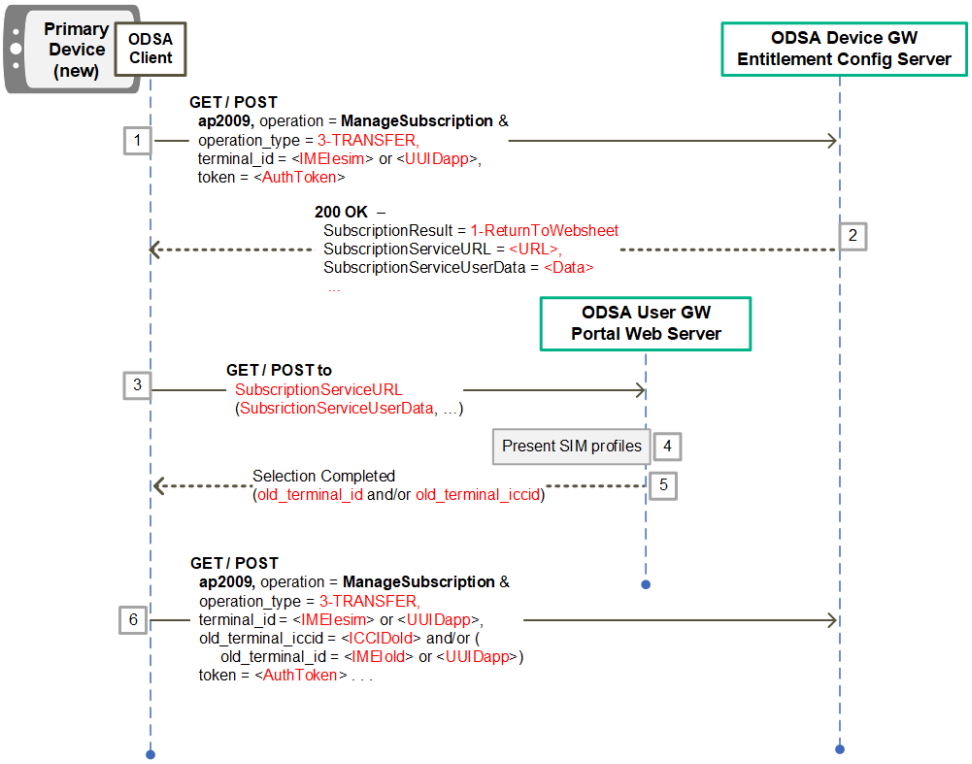

Figure 36. Obtaining the old terminal identifier via web server

8.7 Subscription Transfer with EAP-AKA

The following presents the case where:

- The Primary ODSA device application is allowed for the type of primary device and enabled by the Service Provider (entitled).
- The end-user requests subscription transfer on the old primary device which carries an active subscription with the SP and the end-user is accessible to the eSIM data.
- There is no need to involve the SP's ODSA portal web server as the same type of subscription and plan is activated on the new primary device.
- ECS is capable of handling EAP-AKA relay to a SP's Authentication server (a 3GPP AAA for example).

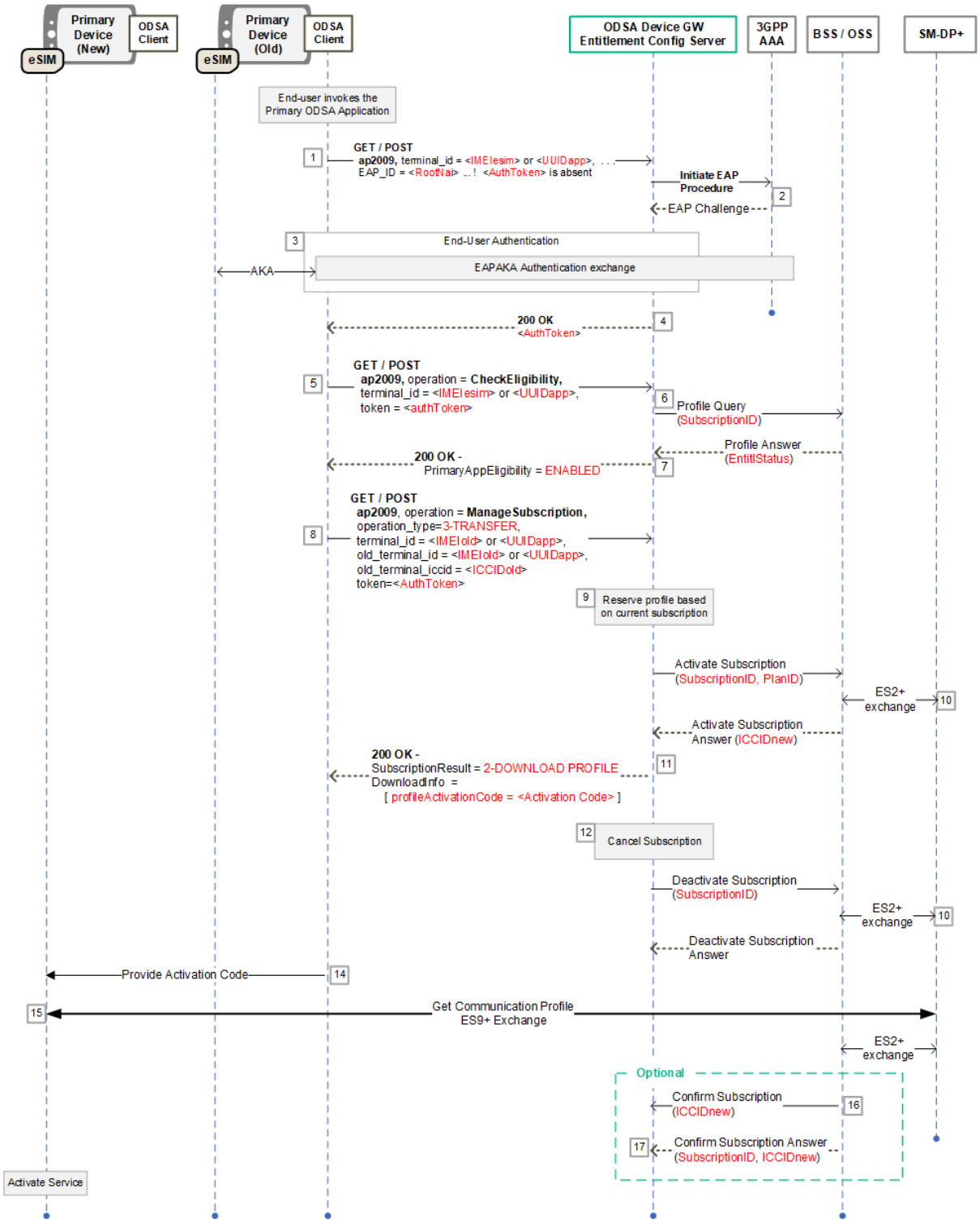

Figure 37 shows the steps of the flow for the activation of a subscription based on an existing subscription validated via EAP-AKA. The Primary ODSA app acquires proper entitlement and subscription data from the SP's ECS. The steps are:

1. User requests On-Device Activation via the Primary ODSA client application. This request may be done by scanning the QR code format of the device information of the new device as defined in section 6.9.1. The client discovers that the end-user

wants to transfer an existing subscription and sends an initial request to the ECS, which includes proper old terminal and EAP_ID parameters.
2. The ECS detects EAP-AKA capability from client, initiates EAP procedure with AuthN server and obtains EAP Challenge.
3. Authentication of the end-user by the SP's 3GPP AAA server is performed using proper EAP-AKA exchanges (see 2.6.1 for details).
4. At the conclusion of the Authentication, the ECS returns new ECS-generated AuthN Token to the ODSA application.
5. The Primary ODSA client application makes a **CheckEligibility** request to the ECS.
6. The ECS queries the SP back-end system managing the entitlements and profile associated with ODSA applications.
7. The ECS generates proper response with application status (ENABLED)
8. The Primary ODSA client application sends a **ManageSubscription** request to the ECS to start the subscription procedure with the SP. If old_terminal_id is present, ECS recognizes that this request is from the old primary device. The request may contain target terminal eid and/or target terminal id.
9. The ECS requests for a new subscription from the SP's back-end system.
10. A set of eSIM profile requests over the ES2+ interface (for example, `DownloadOrder`, `ConfirmOrder` and `ReleaseProfile`) is made to the SM-DP+, resulting in profile download information for the new primary device. If ProfileSmdpAddress parameter of `DownloadInfo` is used, the target_terminal_eid value must be used for profile preparation.
11. The ECS sends subscription information (details of the communication profile) back to the app along with subscription result (2-DOWNLOAD PROFILE).
12. The ECS requests for a subscription cancellation from the SP's back-end system.
13. A set of eSIM profile requests is made to the SM-DP+ to cancel the current subscription.
14. The primary ODSA client application informs the eSIM in the new primary device to download the profile (e.g., QR Code scanning).
15. The new device's eSIM gets the profile from the SM-DP+ via ES9+ channel.
16. Optional - SP’s back-end system requests to make another **ManageSubscription** to the ECS by providing/confirming the Profile’s ICCID.
17. Optional - As a return, The ECS sends proper response. This response includes newly created Subscription ID which is linked to the ICCID in the Subscription Management system.
18. As the new primary device’s subscription and service is in the right state, primary device can initiate cellular service.

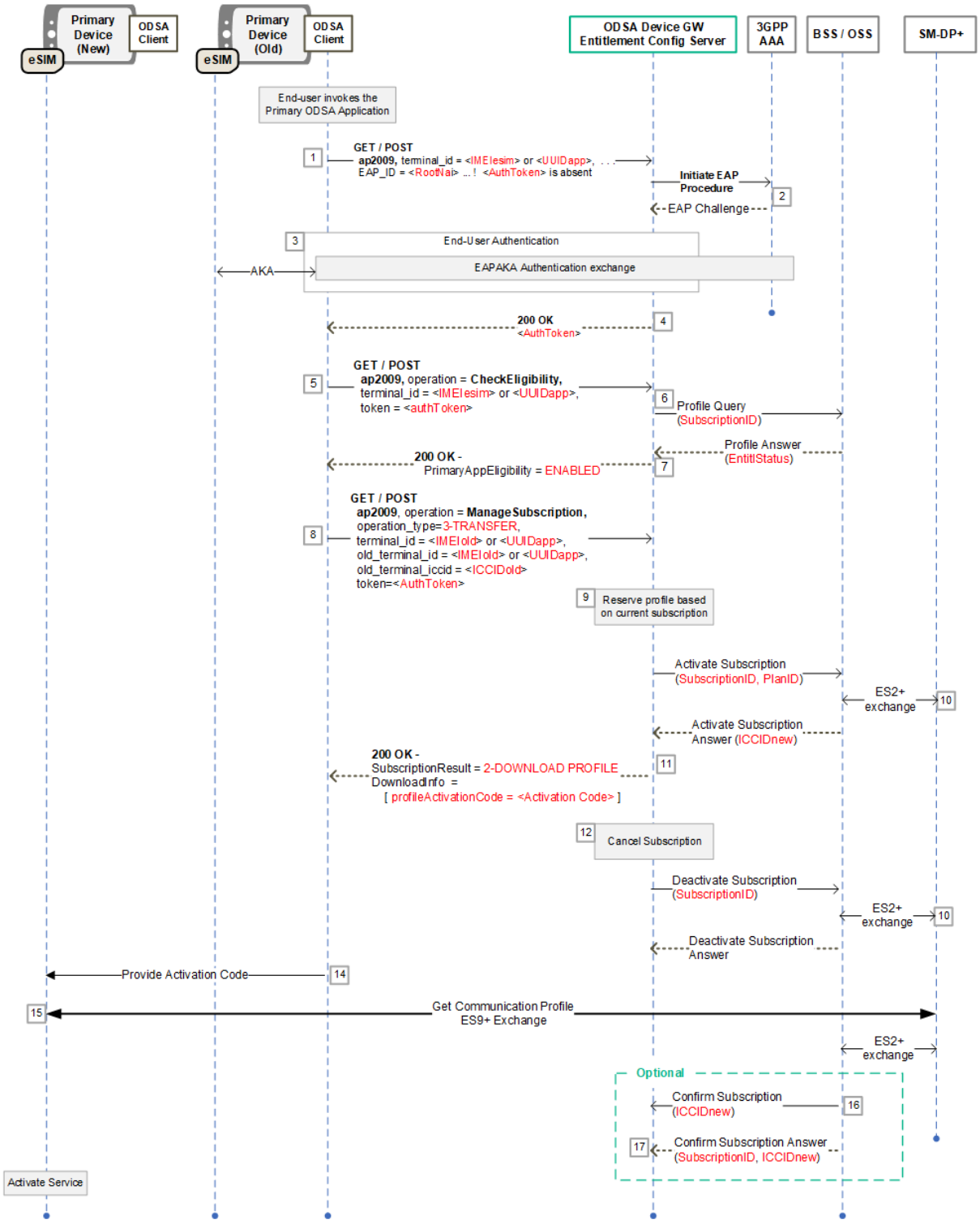

Figure 37: Primary ODSA procedure for Subscription Transfer with EAP-AKA

8.8 Primary ODSA service without ODSA Portal

The following presents the case where:

- The Primary ODSA client application is allowed for the type of primary device and enabled for the end-user (entitled).

- The SP is able to activate or transfer a subscription and create an eSIM profile for the primary device without involving the ODSA portal web server (i.e. native UX is used).
- There is no need to send the end-user to an ODSA portal web server.
- There is one eSIM profile to install or transfer on the primary device.

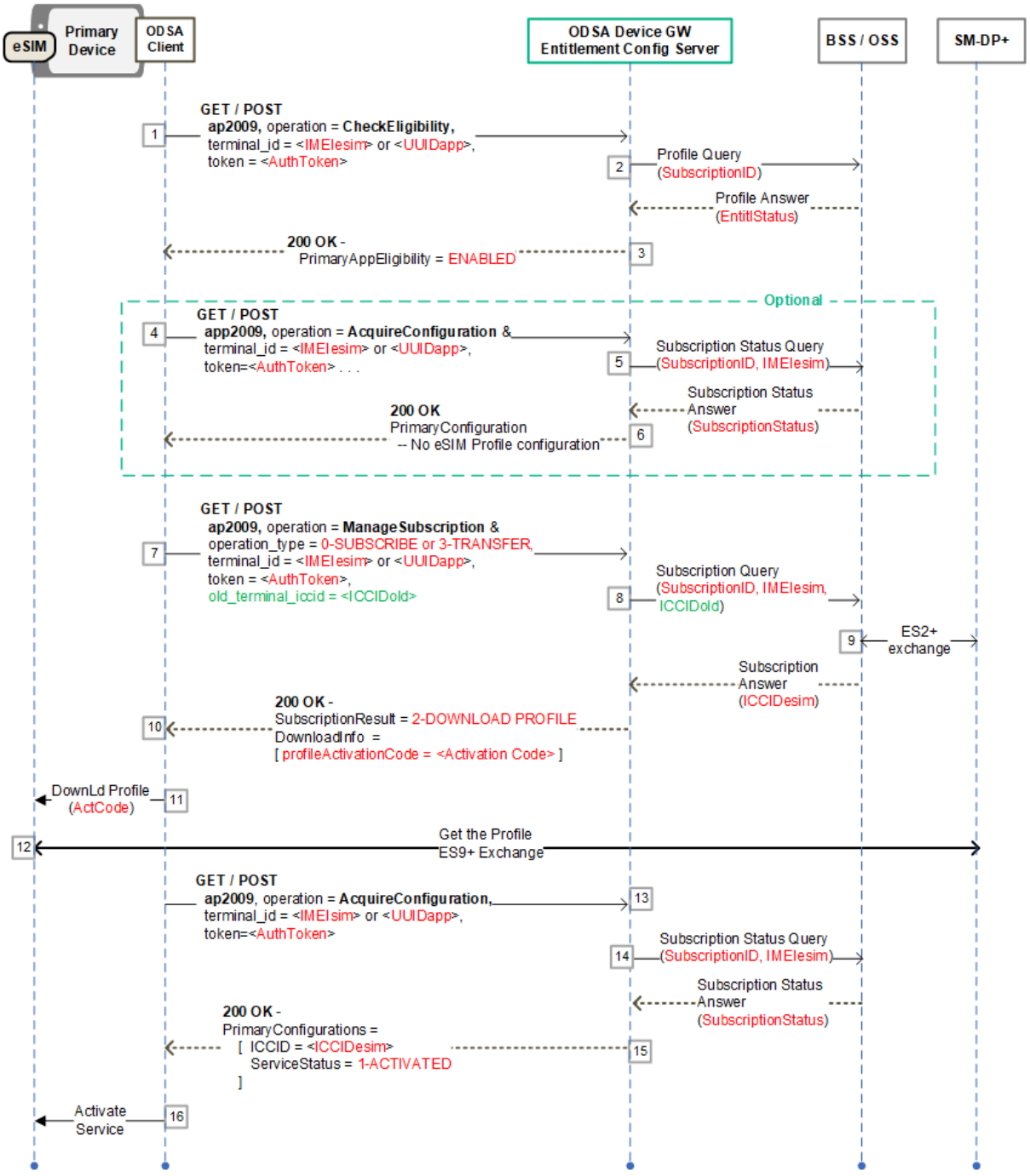

*Figure 38 presents a call flow where the eSIM profile download information for the primary device is made available by the SP at the time of the **ManageSubscription** request. Authentication (e.g. EAP-AKA, SMS-OTP)*

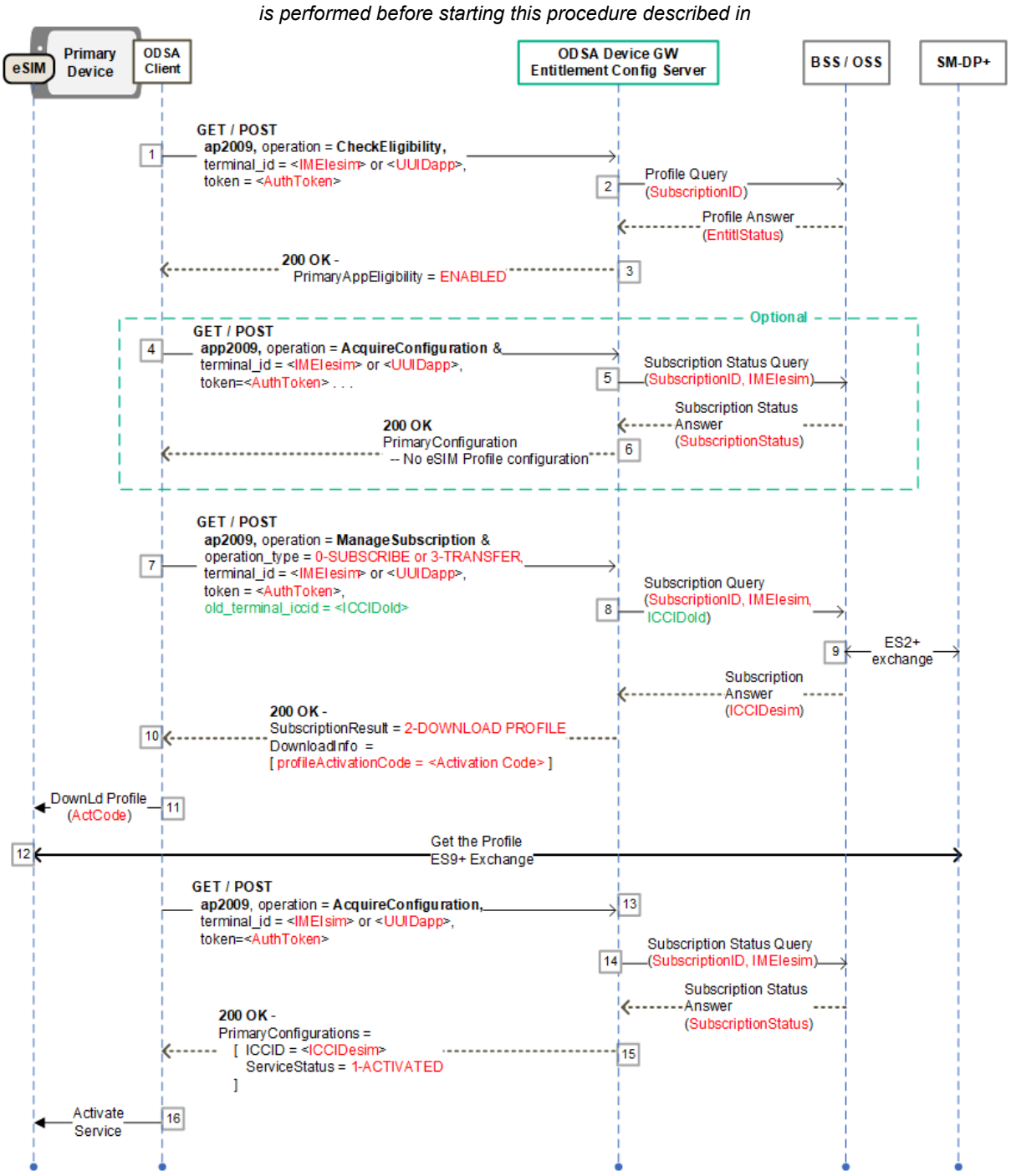

Figure 38.

1. The Primary ODSA client application makes a **CheckEligibility** request to the ECS.
2. The ECS queries the SP back-end system managing the entitlements and eSIM profile associated with the ODSA application.
3. The ECS generates proper response with application status (ENABLED).
4. Optional - Since the target service is allowed, the Primary ODSA application sends an **AcquireConfiguration** request to the ECS to obtain information on the eSIM profile associated with the device.

5. The ECS queries the SP's back-end system managing the subscription and active eSIM profile.
6. The ECS processes the response from the SP's back-end system and generates the proper 200 OK response without any `PrimaryDeviceConfigurations` (no eSIM profile/subscription is associated with the device).
7. The Primary ODSA client application sends a **ManageSubscription** request to the ECS to start the subscription procedure with the SP. It is optional for the device to add `old_terminal_iccid` in the **ManageSubscription** request.
8. The ECS queries the SP's back-end system managing the subscriptions and eSIM profiles. If the `old_terminal_iccid` is available, the ECS checks this value for correctness.
9. The SP's back-end system interacts with the SM-DP+ over the ES2+ interface to make the required eSIM profile requests associated with the new subscription (for example, `DownloadOrder`, `ConfirmOrder` and `ReleaseProfile`) resulting in an activation code and ICCID for the primary device returned to the ECS.
10. The ECS processes the response from the SP's back-end system and generates the proper **ManageSubscription** 200 OK response with a `SubscriptionResult` set to DOWNLOAD_PROFILE (value of 2), and a filled in `DownloadInfo` structure.
11. The primary ODSA client application informs the eSIM to download the eSIM profile.
12. The device's eSIM gets the eSIM profile from the SM-DP+ via ES9+ channel.
13. The ODSA client application makes an **AcquireConfiguration** request to the ECS to verify that the subscription and service for the primary device are in the proper states.
14. The ECS queries the SP's back-end system managing the subscriptions and eSIM profiles.
15. The ECS processes the response from the SP's back-end system and generates the proper 200 OK response containing `PrimaryDeviceConfiguration` with a `PrimaryDeviceConfiguration` entry for the newly active subscription bearing the ACTIVATED status (value of 1).
16. As the primary device’s subscription and service is in the right state, the primary device can initiate cellular service.

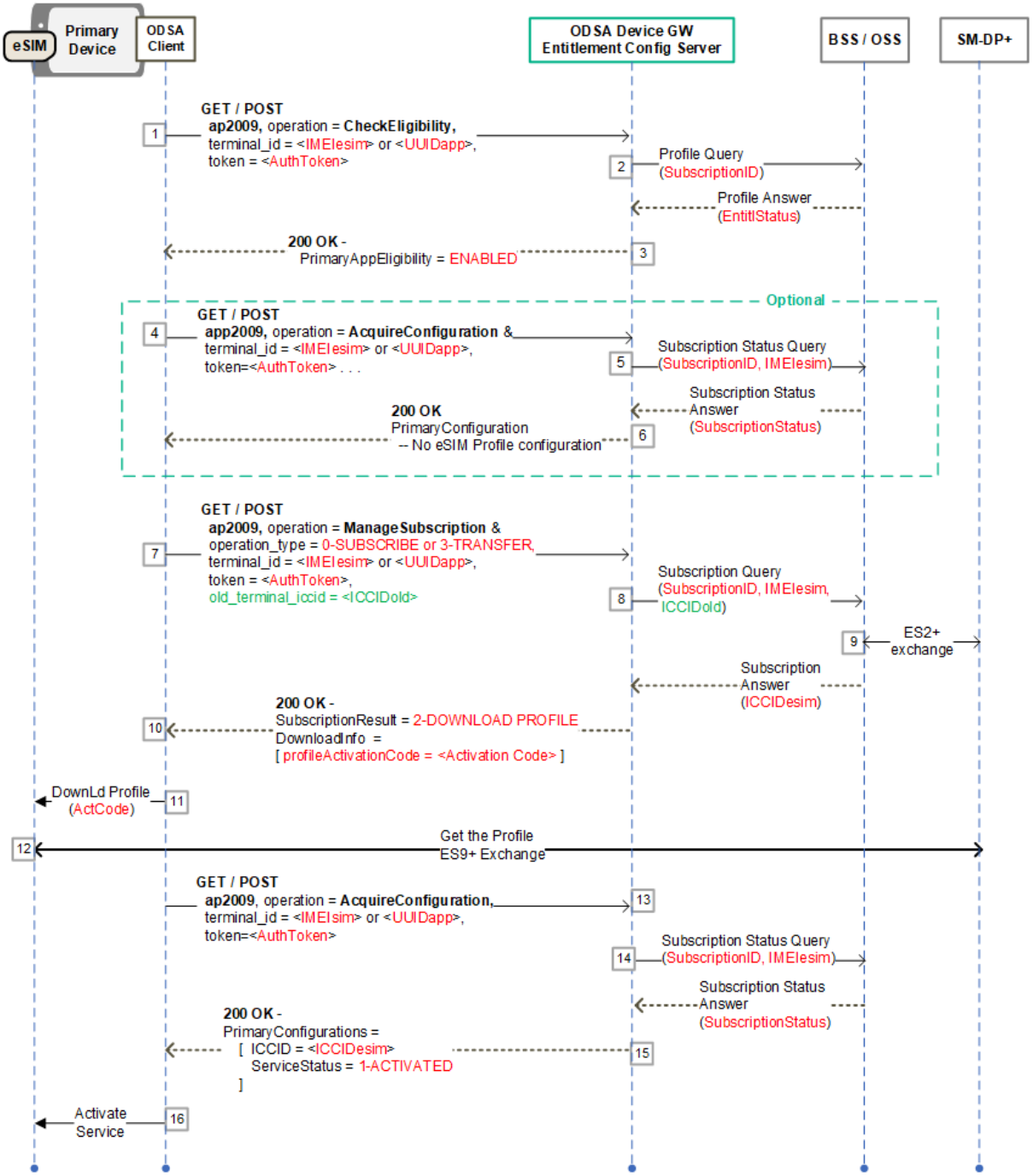

Figure 38: Primary ODSA service when no ODSA Portal is used with immediate download.

Figure 39 presents a call flow where the profile download information for the primary device is not made available by the SP at the time of the **ManageSubscription** request. (delayed delivery).

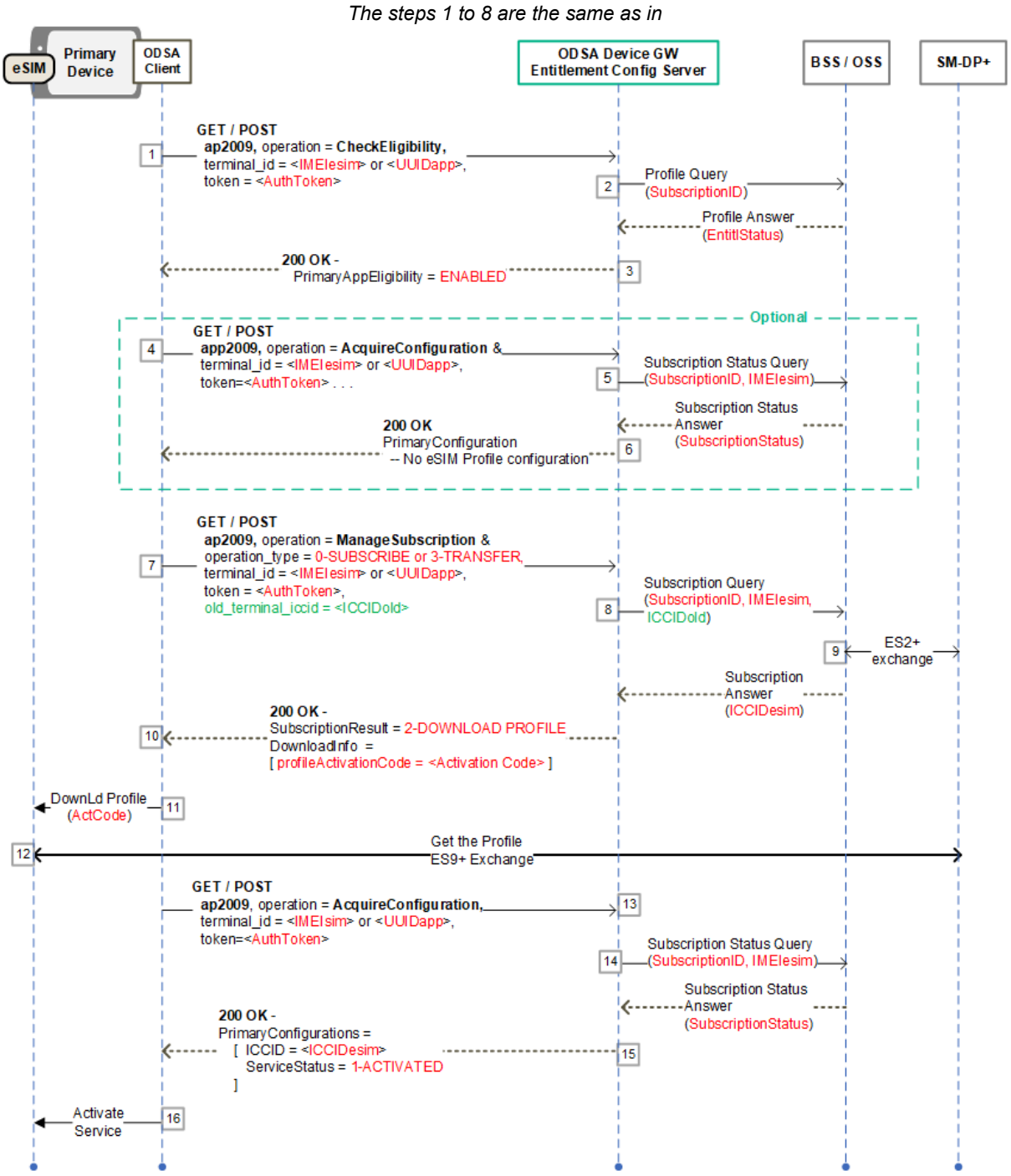

Figure 38. The remaining steps are:

9. The SP's back-end system interacts with the SM-DP+ over the ES2+ interface to make the required eSIM profile requests associated with the subscription and indicates to the ECS that the final response with the download info is delayed (asynchronous).
10. The ECS processes the response from the SP's back-end system and generates the proper **ManageSubscription** 200 OK response with a `SubscriptionResult` set to DELAYED DOWNLOAD (value of 4).

Two different mechanisms can be implemented with this procedure: push and polling. In case of implementing the push mechanism, it should be necessary to follow this step 11 to 16:

11. The ODSA client application makes an **AcquireConfiguration** request to the ECS to verify that the subscription and service for the primary device are in the proper states. ODSA client also adds the `notif_token` and `notif_action` to the request, so that infrastructure-based notifications can be used.
12. The ECS queries the SP's back-end system managing the subscriptions and eSIM profiles. ECS determines that eSIM profile download info is not available, and the subscription is not yet ready.
13. The ECS generates a 200 OK response with a `PrimaryDeviceConfiguration` entry bearing the ACTIVATING status (value of 2). If eSIM profile download info is available in step 12, ECS may send `DownloadInfo` while ACTIVATING. The ODSA client should not expect that receiving `DownloadInfo` while ACTIVATING means `ServiceStatus` is now ACTIVATED. ECS adds the `RegisterNotifStatus` parameter to notify the device about the Notification Registration (0 = SUCCESS).
14. After a delay, as soon as the ECS gets notified about a status change and eSIM profile download info from the MNO-backend, the ECS notifies the ODSA client, using the method defined in `notif_action`.
15. Upon receiving `notif_action`, the ODSA application therefore requests the **AcquireConfiguration**.
16. The ECS queries the SP's back-end system managing the subscriptions and eSIM profiles.

If polling mechanism is implemented, it should be necessary to follow this step 17 to 20 instead of step 11 to 16:

17. The ODSA client application makes an **AcquireConfiguration** request to the ECS to verify that the subscription and service for the primary device are in the proper states.
18. The ECS queries the SP's back-end system managing the subscriptions and eSIM profiles.
    a) If eSIM profile download info is not available and the subscription is not yet ready before reaching the `MaxRefreshRequest`, go to step 19.
    b) If eSIM profile download info is not available and the subscription is not yet ready when to reach the `MaxRefreshRequest`, go to step 20.
    c) If eSIM profile download info is available and the subscription is ready before reaching the `MaxRefreshRequest`, go to step 21.
19. The ECS generates a 200 OK response with a `PrimaryDeviceConfiguration` entry bearing the ACTIVATING status (value of 2). ECS also adds the `PollingInterval`. If eSIM profile download info is available in step 18, ECS may send `DownloadInfo` while ACTIVATING. The ODSA client should not expect that receiving `DownloadInfo` while ACTIVATING means that `ServiceStatus` is now ACTIVATED. ODSA client repeats steps 17 to 19 to check the status update.
20. The ECS returns `PrimaryDeviceConfiguration` bearing the DEACTIVATED, NO REUSE status (value of 4). At this point, the activation flow is finished.

The remaining common steps for both push and polling are:

21. The ECS generates a 200 OK response with a `PrimaryDeviceConfiguration` entry for the newly active subscription bearing the ACTIVATED status (value of 1) and a filled in `DownloadInfo` structure.
22. The primary ODSA client application informs the eSIM to download the eSIM profile.
23. The device's eSIM gets the eSIM profile from the SM-DP+ via ES9+ channel.
24. Both eSIM profile installed and ServiceStatus=Activated are needed to use the service. As the primary device’s subscription and service is in the right state, the primary device can initiate cellular service.

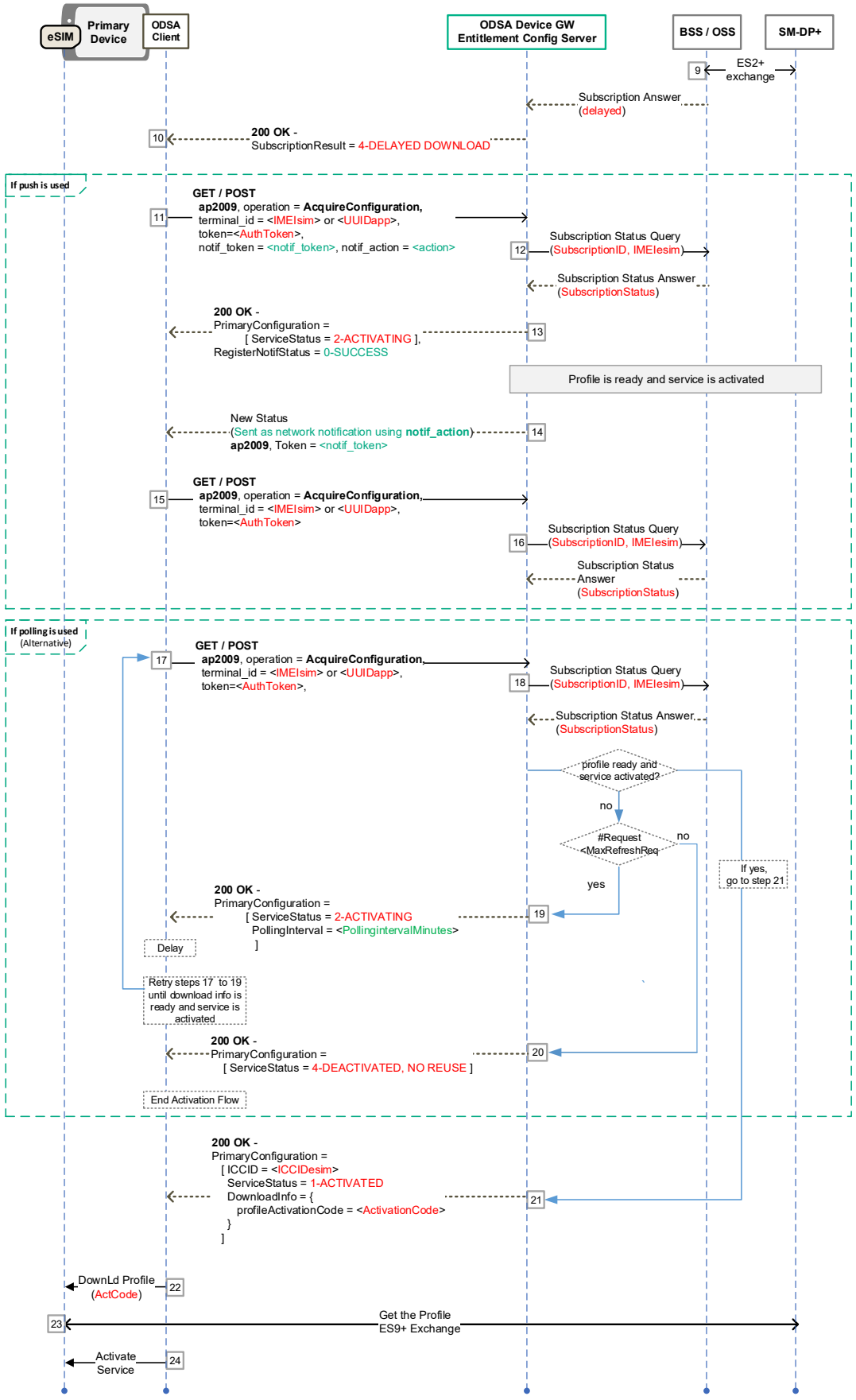

Figure 39: Primary ODSA service when no ODSA Portal is used with delayed download info.

8.9 Subscription Transfer with TemporaryToken

The following presents the case where:

- The Old and New Primary ODSA client applications are allowed for the type of primary device and enabled for the end-user (entitled).
- The New Primary ODSA client application does not have access to the primary profile and its TOKEN to transfer the subscription and uses a temporary token as authentication token to complete the transfer.

Figure 40 presents a call flow where an old device requests a temporary token for use with ManageSubscription and AcquireConfiguration, a new device will use this temporary token to trigger the transfer and the download of the eSIM profile. This download is completed immediately.

1. The **Old Primary** ODSA client application sends a request to the ECS to trigger the authentication procedure (in this case EAP-AKA).
2. ECS and the Old Primary device go through the authentication exchange procedure.
3. The ECS returns a TOKEN to the Old Primary ODSA client application.
4. The Old Primary ODSA client application makes a CheckEligibility request to the ECS. The parameter target_terminal_entitlement_protocol is included when subscription transfer is for cross-TS.43 platform.
5. The ECS queries the SP back-end system managing the entitlements and profile associated with ODSA applications.
6. The ECS generates proper response with application status (ENABLED)
7. The Old Primary ODSA client application requests a temporary token to the ECS and includes the operation_targets to indicate what requests may be used by a trusted third party (in this case the New Primary client application). The parameter target_terminal_entitlement_protocol is included when subscription transfer is for cross-TS.43 platform.
8. The ECS shall respond with a new temporary token if the subscription allows the OperationTargets to be used. In this case, this temporary token can be used in any future (up until the TemporaryTokenExpiry) ManageSubscription and AcquireConfiguration requests from the New Primary ODSA application which doesn't have access to the TOKEN.
9. The Old Primary device shall transmit all relevant eSIM transfer information to the New Primary device. The mechanisms to achieve this are outside the scope of this specification.
10. The **New Primary** ODSA client application sends a ManageSubscription with operation_type 3 - TRANSFER and the old_terminal_iccid request to the ECS using the temporary token. The request shall also contain target_terminal_eid and target_terminal_id parameters. The parameter old_terminal_entitlement_protocol is included when subscription transfer is for cross-TS.43 platform.
    Note: it is also possible for the New Primary ODSA client to request a CheckEligibility request prior to the ManageSubscription to the ECS using the `temporary_token`.
11. STEPS 11-13 ARE OPTIONAL: SKIP to step 14 if user interaction without websheet is not required, if user interaction without websheet is required, the ECS sends a

SubscriptionResult 8 – REQUIRES USER INPUT response to the New Primary ODSA client application and includes a MSG object.

12. When the ODSA client receives Subscription Result 8 – REQUIRES USER INPUT with the MSG parameter, the New Primary ODSA client application display the message of the MSG parameter along with the free text field and the `Accept_btn` button. The user will then enter their response and accept.
13. The New Primary ODSA client application sends a ManageSubscription with operation_type 3 - TRANSFER and the old_terminal_iccid request to the ECS using the temporary token. The New Primary ODSA client SHALL append the user response to the `MSG_response` field and the `MSG_btn` value selected by the user.
14. The ECS queries the SP back-end for a subscription transfer.
15. The SP backend generates a new profile to complete the transfer.
16. The SP back-end provides the ECS with an activation code or new ICCID and SM-DP+ address.
17. The ECS sends a 2 – DOWNLOAD PROFILE response to the Primary ODSA client application and includes the profile activation code or new ICCID and SM-DP+ address. In the case where the ECS wishes to trigger a 4 – DELAYED DOWNLOAD please refer to Figure 40.
18. The New Primary ODSA client application starts the profile download.
19. The New Primary device downloads the profile from the DP+ and the subscription transfer is completed.
20. The New Primary ODSA client application can consider the service activated.

Steps 21-31: Optionally the New Primary ODSA client application and the ECS shall now consider the temporary token expired and the New Primary ODSA application shall use the normal authentication procedures to obtain a TOKEN in order to interact with the ECS.

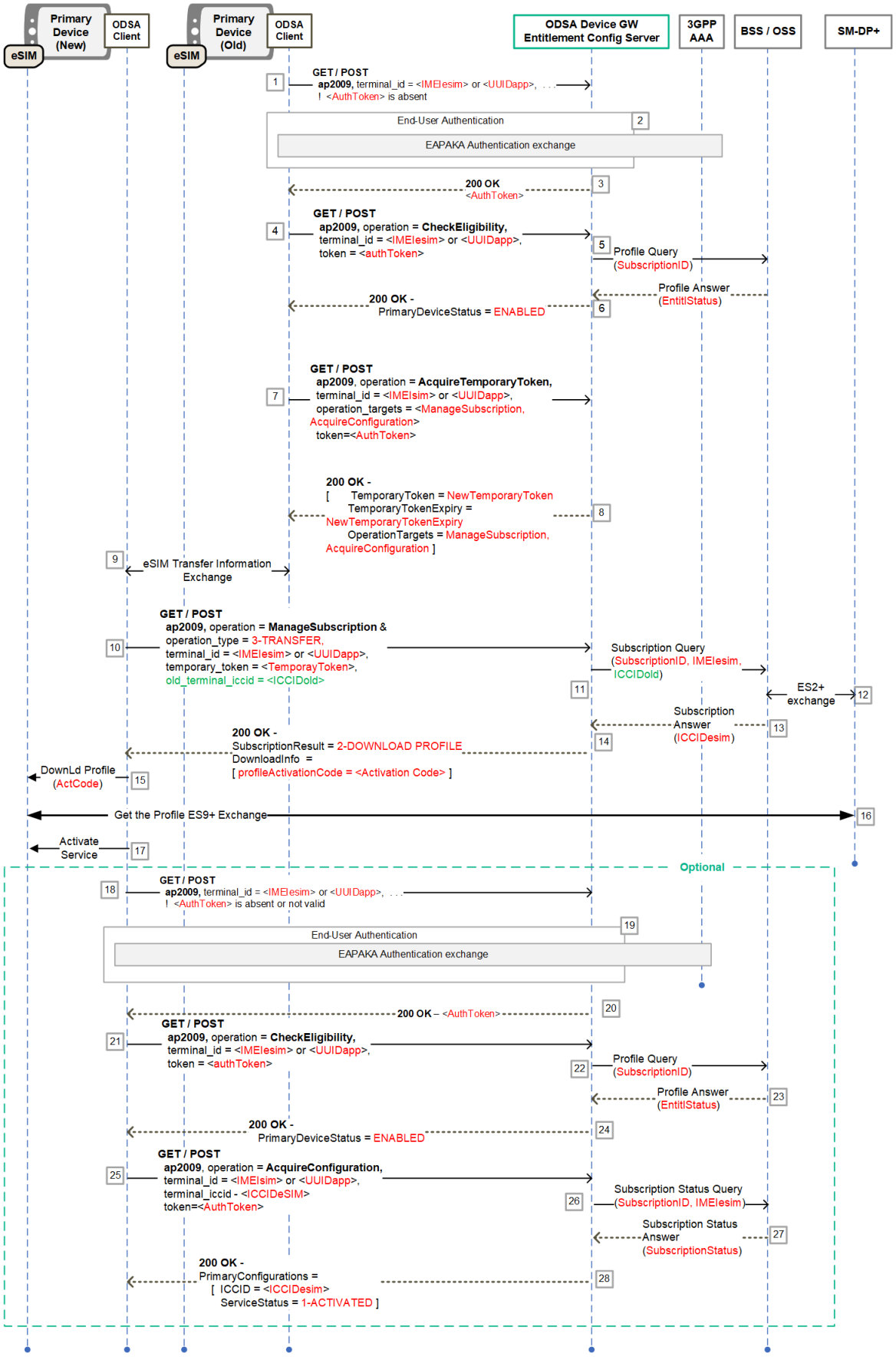

Figure 40: Subscription Transfer Using Temporary Token immediate activation and download.

Figure 41 presents how the temporary token can be used by the New Primary ODSA client application in the cases where there is a delayed activation or download of the eSIM after the transfer request has been sent to the ECS. Note: if there is a delayed activation of the new eSIM, there may be situations where the Old Primary ODSA device loses its subscription and access to the network during this transfer process.

The steps 1 to 15 are the same as in Figure 40. The remaining steps are:

16. The SP's back-end system interacts with the SM-DP+ over the ES2+ interface to make the required eSIM profile requests associated with the subscription and indicates to the ECS that the final response with the download info is delayed (asynchronous).
17. The ECS processes the response from the SP's back-end system and generates the proper **ManageSubscription** 200 OK response with a `SubscriptionResult` set to DELAYED DOWNLOAD (value of 4).

Two different mechanisms can be implemented with this procedure: push and polling. In case of implementing the push mechanism, it should be necessary to follow this step 18 to 23:

18. The New Primary ODSA client application makes an **AcquireConfiguration** request to the ECS using the temporary token to verify that the subscription and service for the primary device are in the proper states. ODSA client also adds the `notif_token` and `notif_action` to the request, so that infrastructure-based notifications can be used. The parameter old_terminal_entitlement_protocol is included when subscription transfer is for cross-TS.43 platform.
19. The ECS queries the SP's back-end system managing the subscriptions and eSIM profiles. ECS determines that eSIM profile download info is not available, and the subscription is not yet ready.
20. The ECS generates a 200 OK response with a `PrimaryDeviceConfiguration` entry bearing the ACTIVATING status (value of 2). If eSIM profile download info is available in step 12, ECS may send `DownloadInfo` while ACTIVATING. The New Primary ODSA client should not expect that receiving `DownloadInfo` while ACTIVATING means `ServiceStatus` is now ACTIVATED. ECS adds the `RegisterNotifStatus` parameter to notify the device about the Notification Registration (0 = SUCCESS).
21. After a delay, as soon as the ECS gets notified about a status change and eSIM profile download info from the MNO-backend, the ECS notifies the New Primary ODSA client, using the method defined in `notif_action`.
22. Upon receiving `notif_action`, the New Primary ODSA client application therefore requests the **AcquireConfiguration**.
23. The ECS queries the SP's back-end system managing the subscriptions and eSIM profiles.

If polling mechanism is implemented, it should be necessary to follow this step 24 to 26 instead of step 18 to 23:

24. The New Primary ODSA client application makes an **AcquireConfiguration** request to the ECS to verify that the subscription and service for the primary device are in the proper states.
25. The ECS queries the SP's back-end system managing the subscriptions and eSIM profiles.
26. If eSIM profile download info is not available and the subscription is not yet ready the ECS generates a 200 OK response with a `PrimaryDeviceConfiguration` entry bearing the ACTIVATING status (value of 2). ECS also adds the `PollingInterval`. If eSIM profile download info is available, ECS may send `DownloadInfo` while ACTIVATING. The ODSA client should not expect that receiving `DownloadInfo` while ACTIVATING means that `ServiceStatus` is now ACTIVATED. ODSA client repeats steps 24 to 26 to check the status update.

The remaining common steps for both push and polling are:

27. The ECS generates a 200 OK response with a `PrimaryDeviceConfiguration` entry for the newly active subscription bearing the ACTIVATED status (value of 1) and a filled in `DownloadInfo` structure.
28. The New Primary ODSA client application informs the eSIM to download the eSIM profile.
29. The device's eSIM gets the eSIM profile from the SM-DP+ via ES9+ channel.
30. Both eSIM profile installed and ServiceStatus=Activated are needed to use the service. As the primary device’s subscription and service is in the right state, the primary device can initiate cellular service.

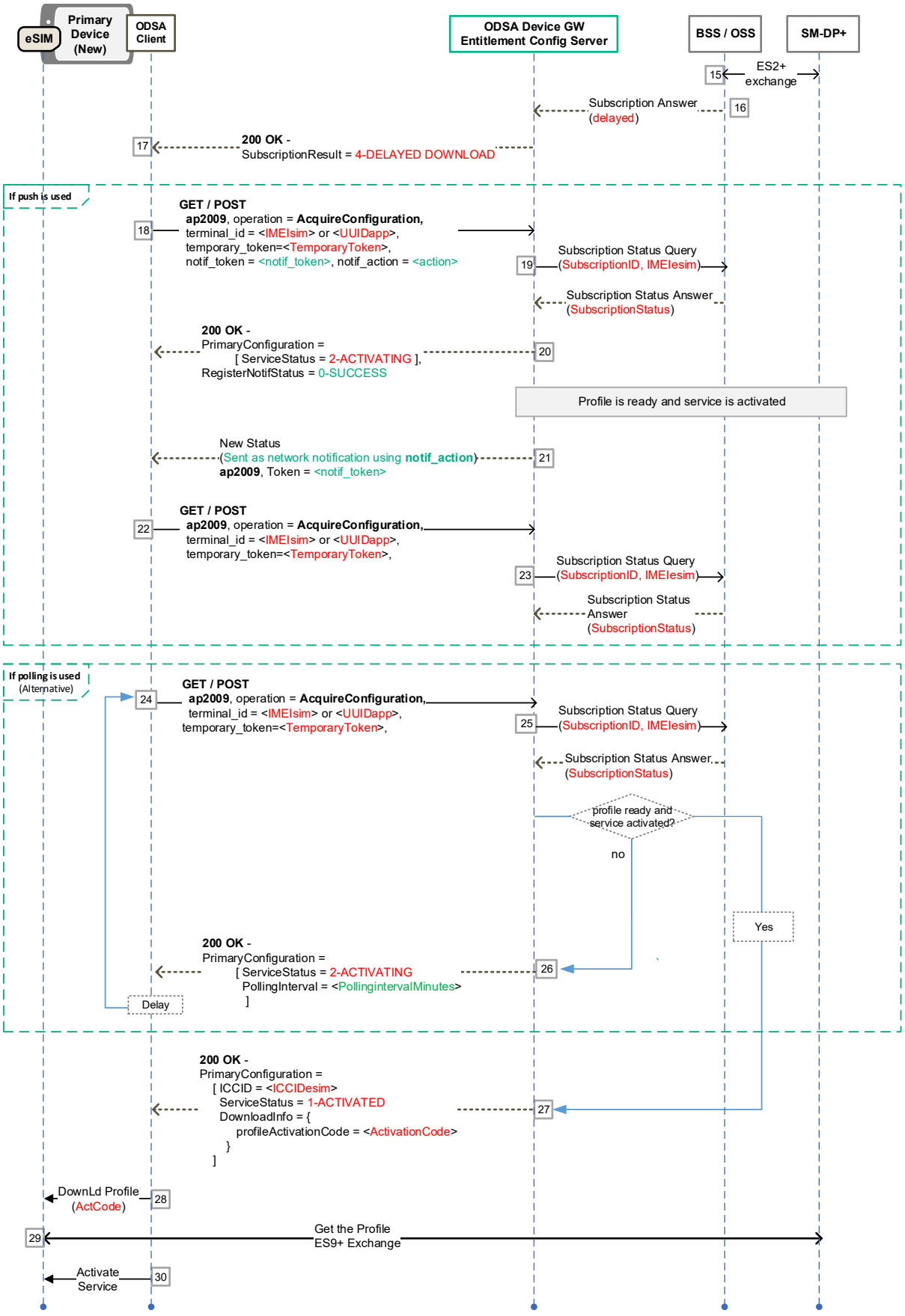

Figure 41: Subscription Transfer Using Temporary Token delayed activation and download.

8.10 VOID

8.11 Subscription Transfer and Deleting Subscription in Old Device

The following presents the case where:

- The Primary ODSA client application is allowed for the type of primary device and enabled for the end-user (entitled).
- The end-user has an active subscription with the SP identified by its MSISDN.
- The SP is able to know if the profile in use needs to be deleted.

8.11.1 Subscription Transfer starting from Old Device without ODSA Portal

The following presents the case where:

- There is no need to involve the SP's ODSA portal web server as the same type of subscription and plan is activated on the new device.
- The SP is able to activate a subscription and create an eSIM profile for the primary device without involving the ODSA portal web server.

Figure 42 presents a call flow where the subscription transfer starts from **old device** and ECS notifies to the device that the profile in use needs to be deleted and then complete the subscription transfer after the user deletes the profile in use.

The steps are:

1. The user requests On-Device Activation via the Primary ODSA client application. The client discovers that the end-user wants to transfer an existing subscription and sends an initial request to the ECS, which includes proper old terminal and EAP_ID parameters.
2. The ECS detects EAP-AKA capability from client, initiates EAP procedure with AuthN server and obtains EAP Challenge.
3. Authentication of the end-user by the SP's 3GPP AAA server is performed using proper EAP-AKA exchanges (see 2.6.1 for details).
4. At the conclusion of the Authentication, the ECS returns new ECS-generated AuthN Token to the ODSA application.
5. The Primary ODSA client application makes a **CheckEligibility** request to the ECS.
6. The ECS queries the SP back-end system managing the entitlements and profile associated with ODSA applications.
7. The ECS generates proper response with application status (ENABLED)
8. The Primary ODSA client application sends a **ManageSubscription** request to the ECS to start the subscription procedure with the SP. If old_terminal_id is present, ECS recognizes that this request is from the **old primary** device.
9. The ECS requests for a new subscription from the SP's back-end system.
10. Check whether deletion operation is needed e.g. A set of eSIM profile requests over the ES2+ interface (for example, `DownloadOrder`, `ConfirmOrder` and `ReleaseProfile`) is made to the SM-DP+, and SM-DP+ recognizes that the profile in use needs to be deleted and then notifies to SP's back-end system.
11. The ECS sends subscription result (6-DELETE PROFILE IN USE) back to the app.

12. The Primary ODSA client application notifies the user that the profile in use needs to be deleted to complete the subscription transfer.
13. When the user deletes the profile in use, **`HandleNotification`** is sent to SM-DP+ over the ES2+ interface.
14. SM-DP+ notifies to SP's backend system that the profile in use has been deleted therefore the subscription transfer can be complete.
15. A set of eSIM profile requests over the ES2+ interface (for example, `DownloadOrder`, `ConfirmOrder` and `ReleaseProfile`) is made to the SM-DP+, resulting in an activation code and ICCID of the profile to be downloaded onto the new primary device.
16. the ECS gets notified about a status change from the MNO-backend.
17. The ECS notifies the old device the ODSA application about a Status Change, using the method defined in `notif_action`.
18. The ODSA client application makes an **AcquireConfiguration** request to the ECS to verify that the subscription and service for the primary device are in the proper states.
19. The ECS queries the SP's back-end system managing the subscriptions and profiles. SP's back-end system notifies the subscription state and eSIM profile download Info.
20. The ECS generates a 200 OK response with a `PrimaryDeviceConfiguration` entry for the newly active subscription bearing the ACTIVATED status (value of 1) and a filled in `DownloadInfo` structure. `ICCIDnew` could be same with `ICCIDold` if the profile is redownloadable.
21. The primary ODSA client application informs the eSIM in the new primary device to download the profile.
22. The device's eSIM gets the profile from the SM-DP+ via ES9+ channel.

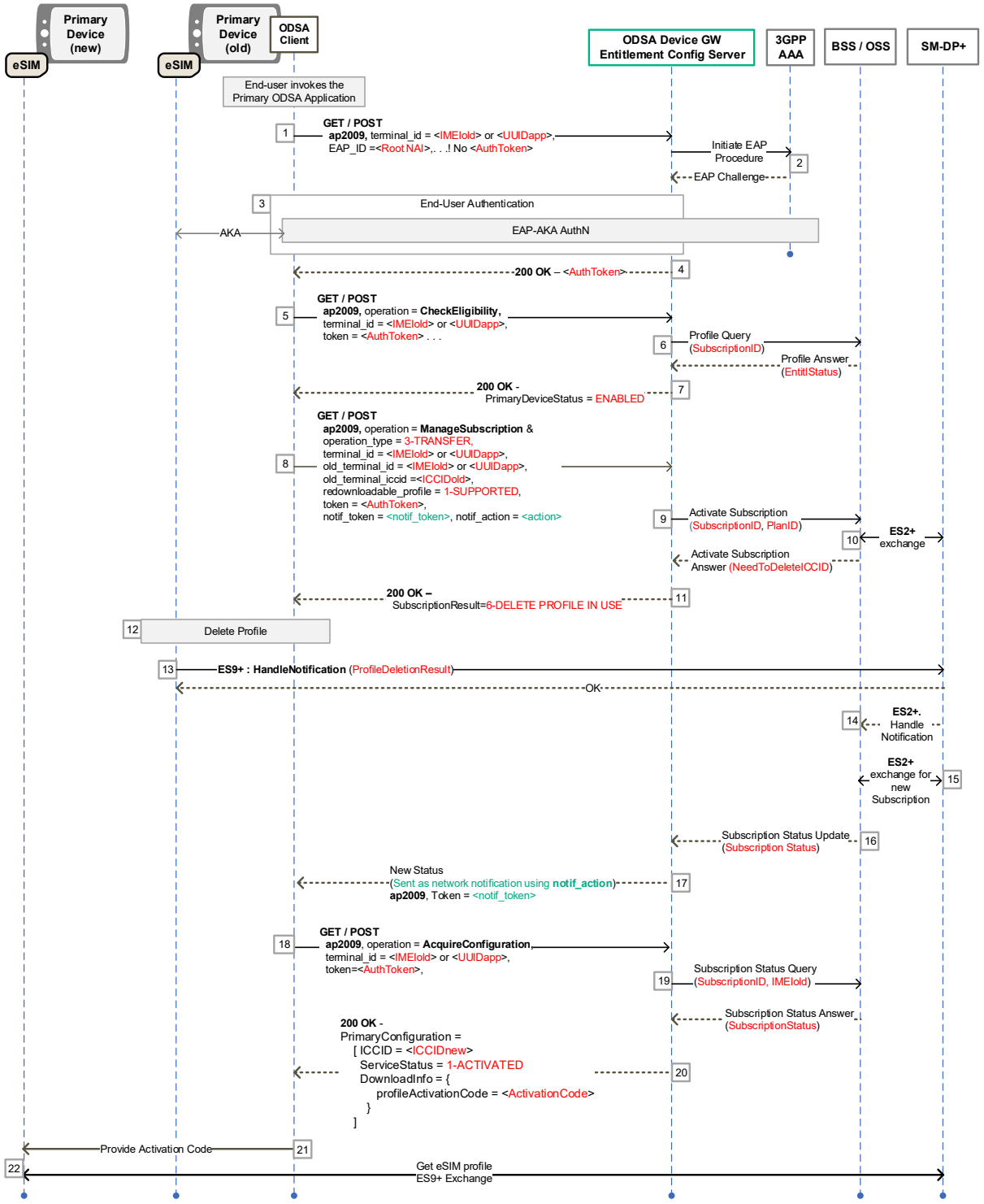

Figure 42. Subscription Transfer starting from Old Device without ODSA Portal

8.11.2 Subscription Transfer starting from New Device via ODSA Portal

The following presents the case where:

- The user normally has an old device since the profile in the old device is required to be deleted. Application can display a pop-up box to ask whether the user has an old phone

or not. However, it depends on applications and is implementation specific. The details are out of scope of this spec.

- The SP's ODSA portal web server is responsible for notifying to the device that the profile in use needs to be deleted before the subscription is transferred.

Figure 43 presents a call flow where the subscription transfer starts from new device and ECS notifies to the device that the profile in use needs to be deleted and then complete the subscription transfer after the user deletes the profile in use.

The steps are:

1. The user requests On-Device Activation via the Primary ODSA client application that sends an initial POST or GET request with proper terminal parameters to the ECS. If SMS-OTP is used, an initial request to the ECS includes MSISDN parameters.
2. The ECS invokes OAuth/OpenID authentication or SMS-OPT authentication which SP supports.
3. At the conclusion of the Authentication, the ECS returns new ECS-generated AuthN Token to the ODSA application.
4. The Primary ODSA client application makes a **CheckEligibility** request to the ECS.
5. The ECS queries the SP back-end system managing the entitlements and profile associated with ODSA applications.
6. The ECS generates proper response with application status (ENABLED)
7. The Primary ODSA client application sends a **ManageSubscription** request to the ECS to start the subscription procedure with the SP.
8. The ECS queries the SP back-end system responsible for managing subscriptions and makes a request for the subscription transfer.
9. The ECS generates a proper response with the subscription procedure data. It contains a `SubscriptionResult` set to CONTINUE_TO_WS (value of 1), and `SubscriptionServiceURL` along with `SubscriptionServiceUserData` presenting the URL of the ODSA Portal web server and any user-specific data that would be useful to the Portal.
10. The Primary ODSA device application sends the end-user to the SP's ODSA web server portal.
11. The SP ODSA portal captures the confirmation on the subscription transfer from the end-user.
12. The SP's back-end system managing subscription receives a new subscription request from the SP portal.
13. A set of eSIM profile requests over the ES2+ interface (for example, `DownloadOrder`, `ConfirmOrder` and `ReleaseProfile`) is made to the SM-DP+, for the new subscription associated with the device eSIM, and SM-DP+ recognizes that the profile in use needs to be deleted and notifies to SP's back-end system.
14. Via a JavaScript call back function, the SP ODSA portal notifies the Primary ODSA app that the profile in use needs to be deleted to complete the subscription transfer.
15. The Primary ODSA device application notifies the user that the profile in use needs to be deleted to complete the subscription transfer.
16. The user deletes the profile in the old device.

17. When the user deletes the profile in use, `HandleNotification` is sent to SM-DP+ over the ES2+ interface.
18. SM-DP+ notifies to SP's backend system that the profile in use has been deleted therefore the subscription transfer can be complete.
19. A set of eSIM profile requests over the ES2+ interface (for example, `DownloadOrder`, `ConfirmOrder` and `ReleaseProfile`) is made to the SM-DP+, resulting in an activation code and ICCID of the profile to be downloaded onto the new primary device.
20. the ECS gets notified about a status change from the MNO-backend.
21. The ECS notifies the ODSA application about a Status Change, using the method defined in `notif_action`.
22. The ODSA client application makes an **AcquireConfiguration** request to the ECS to verify that the subscription and service for the primary device are in the proper states.
23. The ECS queries the SP's back-end system managing the subscriptions and profiles. SP's back-end system notifies the subscription state and eSIM profile download Info.
24. The ECS generates a 200 OK response with a `PrimaryDeviceConfiguration` entry for the newly active subscription bearing the ACTIVATED status (value of 1) and a filled in `DownloadInfo` structure.
25. The device's eSIM gets the profile from the SM-DP+ via ES9+ channel.

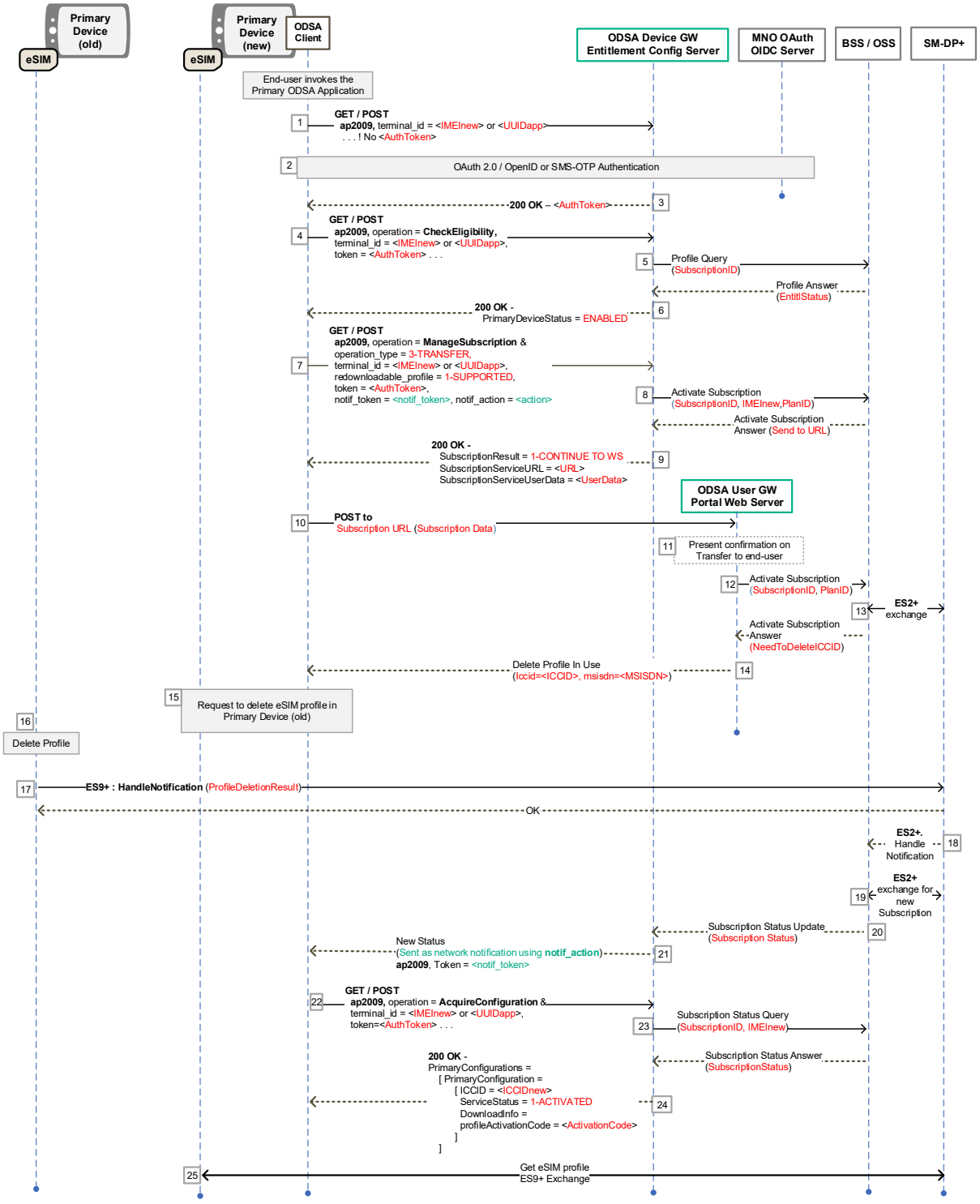

Figure 43. Subscription Transfer starting from New Device via ODSA Portal

8.12 Primary ODSA service requiring user input prior to download on the Default SM-DP+

8.12.1 User verification with ODSA portal

The following presents the case where:

- The user buys a new eSIM device via online channel or offline store.
- The device is shipped to the user and the operator prepares an eSIM profile at their Default SM-DP+ as defined in the SGP.22 [11].
- The operator wants to verify that the user is an eligible user before allowing to download the eSIM profile from the Default SM-DP+.
- The SP's ODSA portal web server is responsible for the user verification.
- The SP prepares one eSIM profile at the Default SM-DP+.

Figure 44 presents the call flow where new eSIM subscription activation with the user verification is handled at the SP's ODSA portal web server, and upon completing the verification successfully, the ECS notifies to the ODSA client that the eSIM profile is released state at the SM-DP+.

1. The Primary ODSA application sends an AcquireConfiguration request to the ECS to get information about the eSIM profile associated with the device.
2. The ECS queries the SP's back-end system managing the subscriptions and the eSIM profile associated with the ODSA applications.
3. The ECS processes the response from the SP's back-end system and generates the proper 200 OK response with Primary Configuration indicating that an eSIM profile/subscription is associated with the device, but the subscription associated with the eSIM profile is not activated (i.e. ServiceStatus=3-DEACTIVATED).
4. When the ODSA client receives ServiceStatus=3-DEACTIVATED, the Primary ODSA client application sends a **ManageSubscription** with operation_type= 0-SUBSCRIBE request to the ECS to start the subscription procedure with the SP. The Primary ODSA client SHALL add terminal_iccid that is returned by the Primary Configuration in step 3.
5. The ECS queries the SP back-end system responsible for managing subscriptions and detects that further end user confirmation is needed before the subscription activation.
6. The ECS generates a proper response with the subscription procedure data. It contains a SubscriptionResult set to **CONTINUE_TO_WS** (value of 1), and SubscriptionServiceURL along with SubscriptionServiceUserData presenting the URL of the ODSA portal web server and any user-specific data that would be useful.
7. The Primary ODSA client application sends the end-user to the SP's ODSA web server portal.
8. The SP's ODSA portal performs the user verification and may capture the confirmation on the subscription activation from the user.
9. The SP's back-end system managing subscription receives a subscription request from the SP's ODSA web portal.
10. In case the eSIM profile is not released at the SM-DP+ server, the SP's back-end system interacts with the Default SM-DP+ over the ES2+ interface to make the required eSIM profile associated with the new subscription is to be released state at the SM-DP+.
11. If immediate download procedure, step 11 is performed:
    Upon the eSIM profile is released, the SP's back-end system notifies it to the SP's ODSA web portal. And then, the SP's ODSA web portal notifies the Default SM-DP address and ICCID to the Primary ODSA client application via a JavaScript call back

function. The ECS MAY be also notified the change from the SP's back-end system to update the subscription status.

12. If delayed procedure, this step 12, and the following step 13 are performed.

    The eSIM profile status change may take some time in SP's back-end systems. If the procedure is delayed, the SP's back-end system notifies it to the SP's ODSA web portal. The SP's ODSA web portal returns the finish flow (no download info) to the Primary ODSA client application.

13. Polling or push notification mechanisms should be implemented for user experience:

    - In case of polling mechanism is used: it is necessary to include the loop for refreshing status as described in the section 7.3.2.
    - In case of push notification is used: ODSA client may request a push notification by registering a push token as described in the section 7.3.1. Alternatively, if agreed between operator and device vendor, the device may use SM-DS push mechanism defined in the SGP.22 [11], which is out of this TS.43 scope.

    Which polling or push notification mechanism a device vendor chooses is up to the agreement between a Service Provider and a device vendor.

14. The Primary ODSA client application sends the user to download the eSIM profile that is prepared at the Default SM-DP+.
15. The device's eSIM gets the eSIM profile from the SM-DP+ via ES9+ channel.

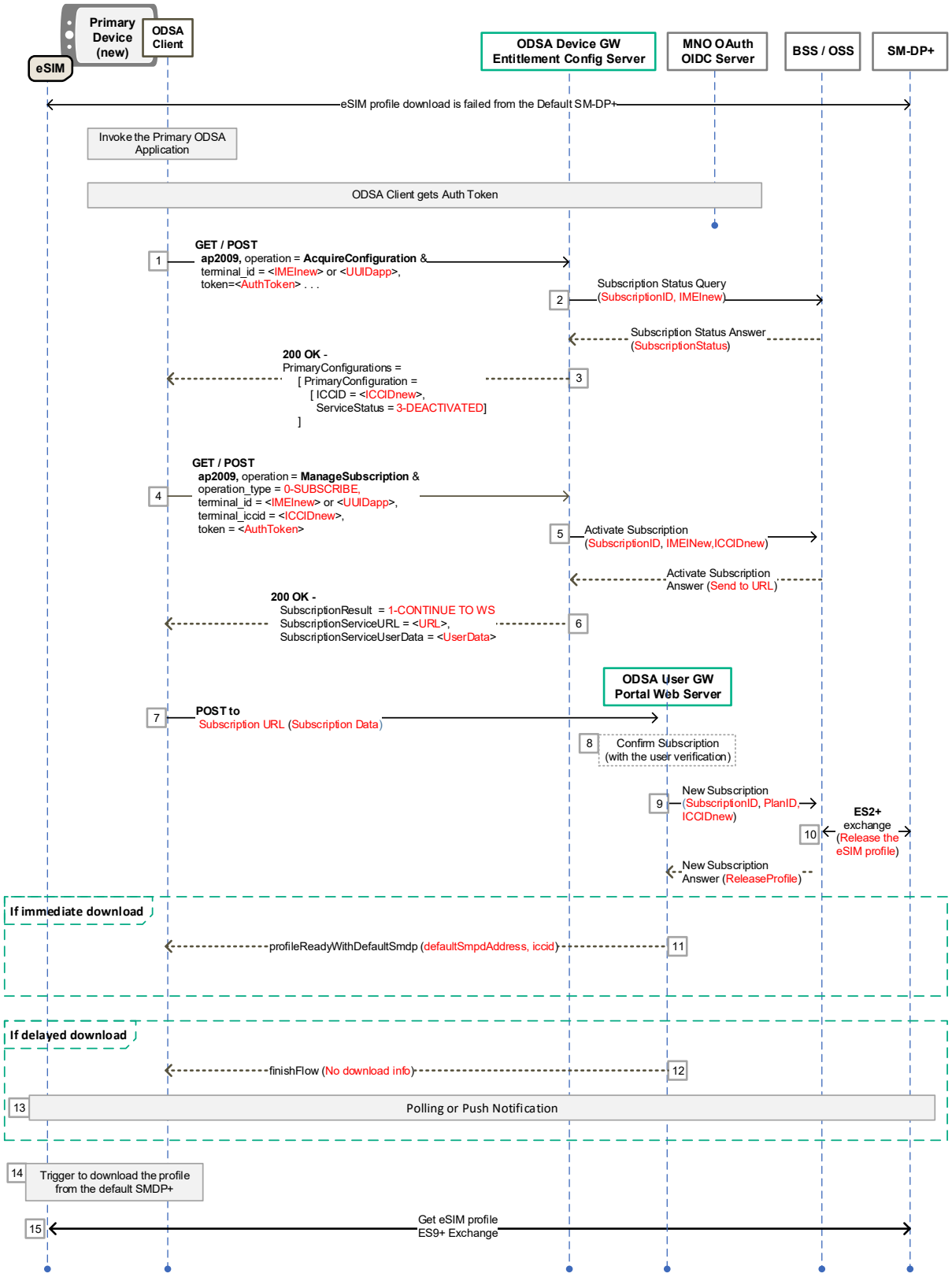

Figure 44. Primary ODSA service requiring user input prior to download on the Default SM-DP+ with ODSA portal.

8.12.2 User verification without an ODSA portal

The following presents the case where:

- The user buys a new eSIM device via online channel or offline store.
- The device is shipped to the user and the operator prepares an eSIM profile at their Default SM-DP+ as defined in the SGP.22 [11].
- The operator wants to verify that the user is an eligible user before allowing to download the eSIM profile from the Default SM-DP+.
- The SP's ODSA ECS combined with OSDA client are responsible for the user verification.
- The SP prepares one eSIM profile at the Default SM-DP+.

Figure 45 presents the call flow where new eSIM subscription activation with the user verification is handled without an ODSA portal web server.

1. The Primary ODSA application sends an AcquireConfiguration request to the ECS to get information about the eSIM profile associated with the device.
2. The ECS queries the SP's back-end system managing the subscriptions and the eSIM profile associated with the ODSA applications.
3. The ECS processes the response from the SP's back-end system and generates the proper 200 OK response with `PrimaryConfiguration` indicating that an eSIM profile/subscription is associated with the device, but the eSIM profile's IMSI is not activated (i.e. ServiceStatus=3-DEACTIVATED). It can return along with the MSG parameter containing the user verification question, a request for a free text field and a `MSG_btn` 'accept' in order to allow the user to acknowledge their response.
4. When the ODSA client receives ServiceStatus=3-DEACTIVATED with the MSG parameter, the Primary ODSA client application displays the message of the MSG parameter along with the free text field and the Accept_btn button. The user will then enter their response and accept.
5. The Primary ODSA client SHALL append the user response to the `MSG_response` field and the `MSG_btn` value selected by the user and add terminal_iccid that is returned by the `PrimaryConfiguration` in step 3 in the ManageSubscription request and send an operation_type 0 SUBSCRIBE.
6. In the case where the `MSG_response` is validated by the ECS, the ECS queries the SP back-end system responsible for managing subscriptions and makes a request in the operator backend for the next steps in the transfer (e.g. SIM SWAP request).
   In the case where the `MSG_response` cannot be not validated by the ECS, go to step 11.
7. If immediate download procedure: The SP's back-end system interacts with the Default SM-DP+ over the ES2+ interface to make the required eSIM profile associated with the new subscription is to be released state at the SM-DP+.
8. If the eSIM profile is ready for immediate download, the ECS generates a proper response with the subscription procedure data. It contains a SubscriptionResult set to **2 - DOWNLOAD PROFILE** including the ProfileSmdpAddress (or ProfileActivationCode), and the flow moves directly to step 12.
9. If delayed eSIM profile download procedure, this step 9, and the following step 10, 12, 13 are performed.

10. Polling or push notification mechanisms should be implemented for user experience:
    - In case of polling mechanism is used: it is necessary to include the loop for refreshing status as described in the section 7.3.2.
    - In case of push notification is used: ODSA client may request a push notification by registering a push token as described in the section 7.3.1. Alternatively, if agreed between operator and device vendor, the device may use SM-DS push mechanism defined in the SGP.22 [11], which is out of this TS.43 scope.
11. If the user input is invalid, the ECS returns and `OperationalResult` **104 – ERROR, INVALID MSG RESPONSE** to inform the client that the response was not accepted by the ECS, and the process ends here.
12. The Primary ODSA client application sends the user to download the eSIM profile that is prepared at the Default SM-DP+.
13. The device's eSIM gets the eSIM profile from the SM-DP+ via ES9+ channel.

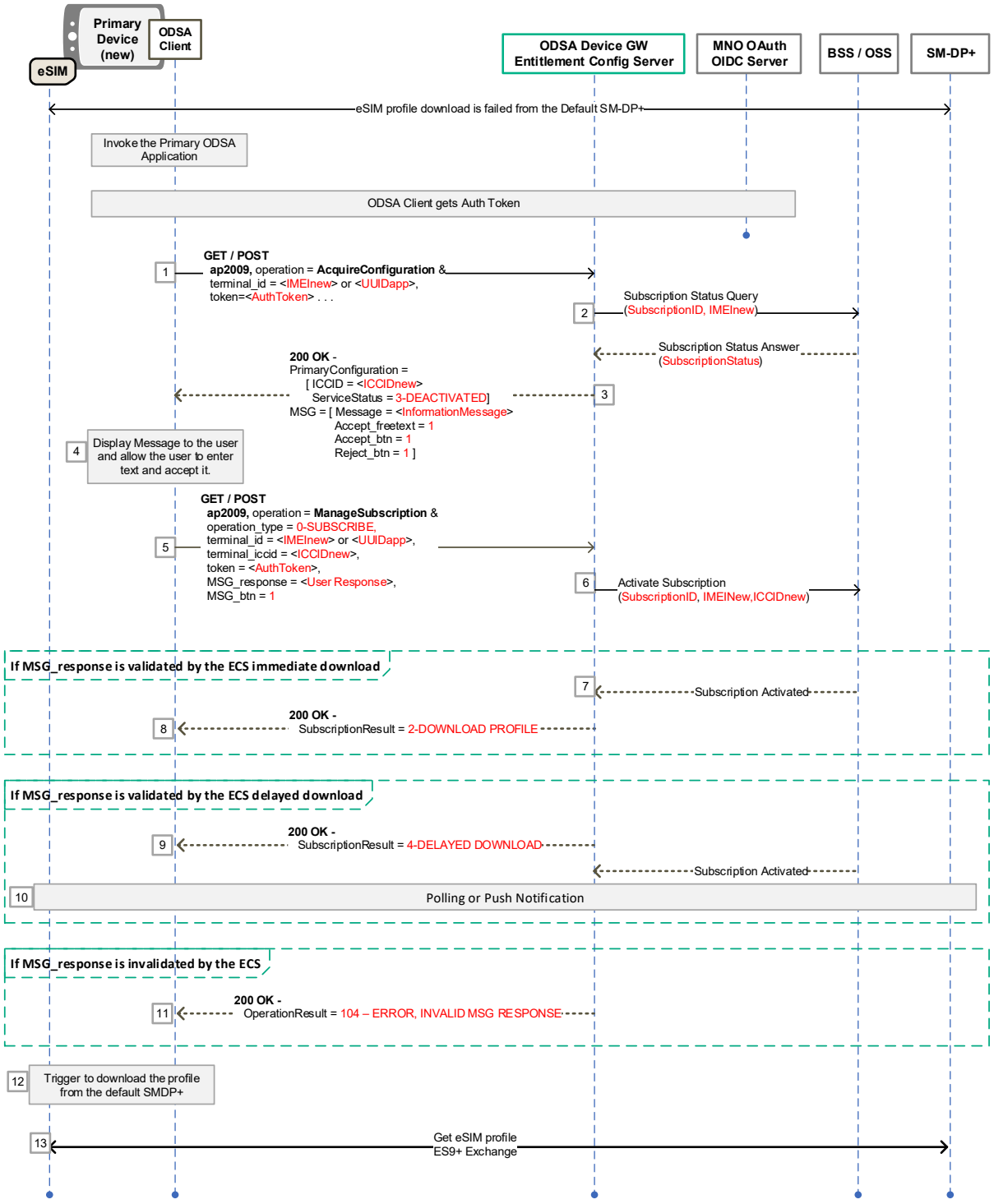

Figure 45. Primary ODSA service requiring user input prior to download on the Default SM-DP+ without ODSA portal.

8.13 Active Subscription Recovery

8.13.1 Active Subscription Recovery with ODSA Portal.

The following presents the case where:

- End user has a primary eSIM device with the active subscription.

- End user factory reset the primary eSIM device or manually remove the existing eSIM profile and the subscription cannot be used at the primary eSIM device any longer.

- End user would like to recover the original subscription via the specific process provide by the SP via the web portal different from the first-time activation.

Figure 46 displays the call flow in which the end user starts the active subscription recovery case, and the SP redirects the end user to a specific web portal URL to complete the subscription recovery process.

1. User requests active subscription recovery procedure via the Primary ODSA client application that sends an initial POST or GET request with proper terminal parameters to the ECS.
2. If there is no parameter associated with authentication or identification (no IMSI in the request), the ECS invokes OAuth/OpenID authentication and connects the app/end-user with the SP's OpenID/OAuth 2.0 platform. If there's other SIM/eSIM profile on the device, SIM based authentication is also possible. This procedure is authentication mode agnostic.
3. At the conclusion of the Authentication, the ECS-generated AuthN Token to the ODSA application.
4. The Primary ODSA client application makes a **CheckEligibility** request to the ECS.
5. The ECS queries the SP back-end system managing the entitlements and profile associated with ODSA applications.
6. The ECS generates proper response with application status (ENABLED)
7. Since the target service is allowed, the Primary ODSA application sends an **AcquireConfiguration** request to the ECS to obtain the original eSIM profiles associated with the device.
8. The ECS queries the SP's back-end system managing the subscriptions for the active profiles.
9. The ECS processes the response from the SP's back-end system and generates the proper 200 OK response with the PrimaryDeviceConfigurations (the original profile/subscription associated to the primary device). The Primary ODSA application identify the "*terminal_iccid*" can be recovered by cross checking the ICCIDs returned by ECS and the ones which are still installed on the primary device.
10. The Primary ODSA client application displays the list of ICCID/Subscriptions can be recovered to the end user and based on end user selection, the primary ODSA client application sends a **ManageSubscription** (Operation= 7 Active Subscription Recover) request with the specific ICCID that has been removed (*terminal_iccid*) to the ECS to start the subscription recovery procedure. Based on the number of the ICCID/Subscription(s) that can be recovered, steps from #10 can be repeated.
11. The ECS queries the SP back-end system responsible for managing subscriptions and makes a request for existing subscription recovery. SP can double check at the SM-DP+ to confirm that if the ICCID to be recovered is really removed from device.
12. The ECS generates a proper response with the subscription procedure data. It contains a SubscriptionResult set to CONTINUE_TO_WS (value of 1), and SubscriptionServiceURL along with SubscriptionServiceUserData presenting the

URL of the ODSA Portal web server and any user-specific data that would be useful to the Portal dedicated for the subscription recovery.

13. The Primary ODSA device application redirects the end-user to the SP's ODSA web server portal for active subscription recovery.
14. The SP ODSA portal shows the details of the original subscription whose eSIM profile has been removed from the device and asks for the final confirmation from the end user.
15. The SP's back-end system managing subscription receives the subscription recovery request from the SP portal.
16. A set of eSIM profile requests over the ES2+ interface (for example, DownloadOrder, ConfirmOrder and ReleaseProfile) is made to the SM-DP+, for the new eSIM profile associated with the device eSIM, resulting in an activation code and ICCID for the primary device.
17. Via a JavaScript call back function, the SP ODSA portal sends subscription information (details of the eSIM profile) back to the Primary ODSA app.
18. The Primary ODSA device application informs the eSIM module to download the eSIM profile, which is obtained from the SM-DP+.
19. The device's eSIM gets the eSIM profile from the SM-DP+ via ES9+ channel.
20. Optional - The Primary ODSA app makes another **ManageSubscription** to the ECS to provide/confirm the download of the newly created ICCID and to validate that the primary device subscription is ready and in proper activated state.
21. The ECS queries the Subscription Management system.
22. The ECS generates the proper response with subscription result (3-DONE).
23. Optional - The Primary ODSA client application makes an **AcquireConfiguration** request to the ECS to verify that the subscription and service for the device are in the proper states.
24. The ECS queries the SP's back-end system managing the subscriptions and profiles.
25. The ECS processes the response from the SP's back-end system and generates the proper 200 OK response containing a PrimaryConfiguration entry for the newly recovered subscription bearing the ACTIVATED status (value of 1).
26. As the primary device’s subscription and service is in the right state, the primary device can initiate cellular service.

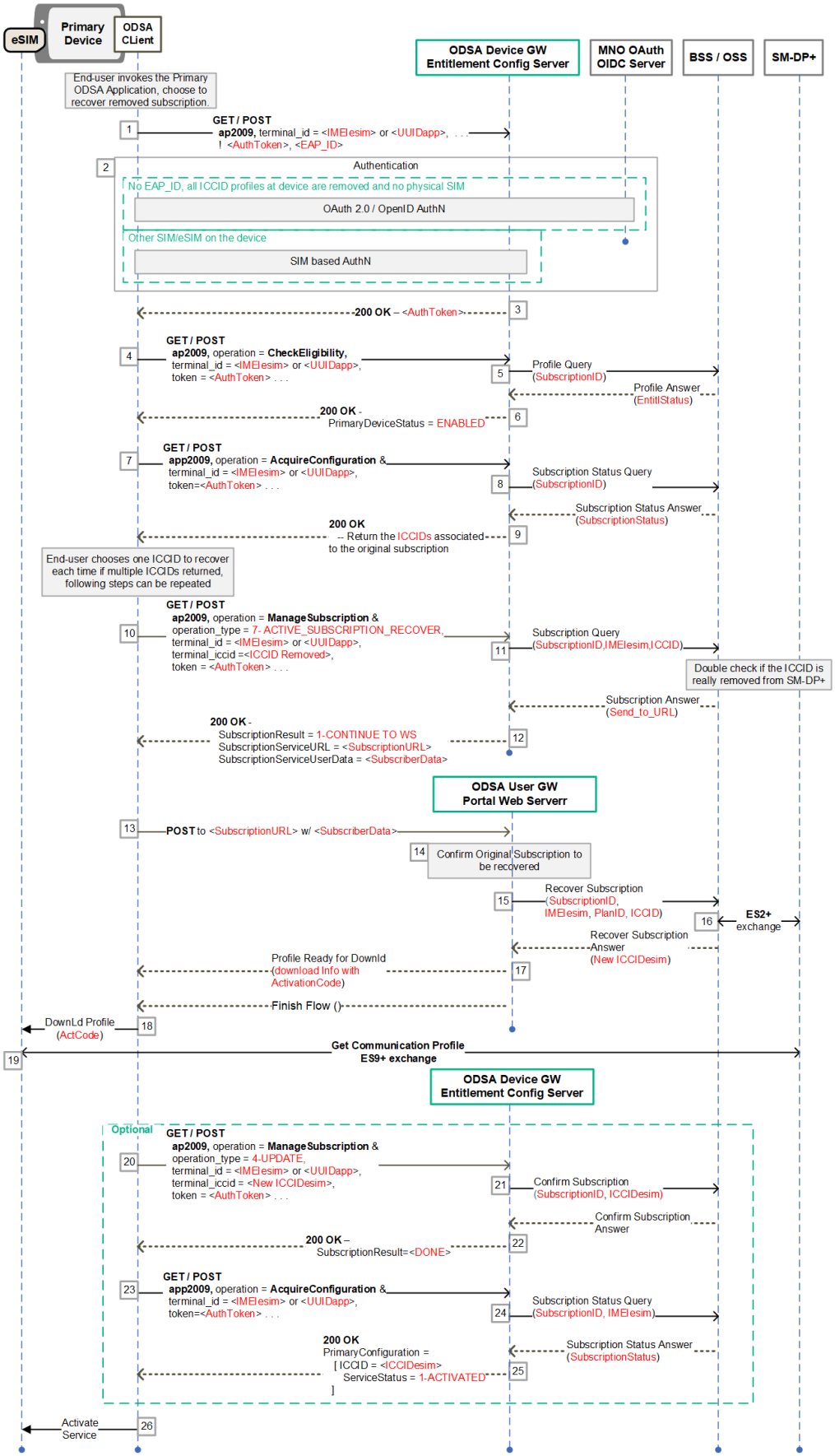

Figure 46. End user request for recovering the subscription which has been removed via ODSA web portal.

8.13.2 Active Subscription Recovery without ODSA Portal.

The following presents the case where:

- End user has a primary eSIM device with the active subscription.
- End user factory reset the primary eSIM device or manually remove the existing eSIM profile and the subscription cannot be used at the primary eSIM device any longer.
- End user would like to recover the original subscription via the native UI, no need to redirect the end user to the web portal.

Figure 47 presents the call flow that the end user starts the active subscription recovery case, SP indicate end user to complete the subscription recovery process at native UI directly.

1. User requests active subscription recovery procedure via the Primary ODSA client application that sends an initial POST or GET request with proper terminal parameters to the ECS.
2. If there is no parameter associated with authentication or identification (no IMSI in the request), the ECS invokes OAuth/OpenID authentication and connects the app/end-user with the SP's OpenID/OAuth 2.0 platform. If there's other SIM/eSIM profile on the device, SIM based authentication is also possible. This procedure is authentication mode agnostic.
3. At the conclusion of the Authentication, the ECS-generated AuthN Token to the ODSA application.
4. The Primary ODSA client application makes a **CheckEligibility** request to the ECS.
5. The ECS queries the SP back-end system managing the entitlements and profile associated with ODSA applications.
6. The ECS generates proper response with application status (ENABLED)
7. Since the target service is allowed, the Primary ODSA application sends an **AcquireConfiguration** request to the ECS to obtain the original eSIM profiles associated with the device.
8. The ECS queries the SP's back-end system managing the subscriptions and active profiles.
9. The ECS processes the response from the SP's back-end system and generates the proper 200 OK response with the PrimaryDeviceConfigurations (the original profile/subscription associated to the primary device). The Primary ODSA application identify the "*terminal_iccid*" can be recovered by cross checking the ICCIDs returned by ECS and the ones which are still installed on the primary device.
10. The Primary ODSA client application displays the list of ICCID/Subscriptions can be recovered to the end user and based on end user selection, the primary ODSA client application sends a **ManageSubscription** (Operation= 7 Active Subscription Recover) request with the specific ICCID that has been removed (*terminal_iccid*) to the ECS to start the subscription recovery procedure. Based on the number of the ICCID/Subscription(s) that can be recovered, steps from #10 can be repeated.

11. The ECS queries the SP back-end system responsible for managing subscriptions and makes a request for existing subscription recovery. SP can double check at the SM-DP+ to confirm that if the ICCID to be recovered is really removed from device.
12. Optional - The ECS detects that the removed subscription recovery operation requires further end user interaction and sends a SubscriptionResult 8 – REQUIRES USER INPUT response to the New Primary ODSA client application and includes a MSG object.
13. Optional - The Primary ODSA device application instructs the end user based on the content in the MSG to complete the interaction and sends another **ManageSubscription** (Operation= 7 Active Subscription Recover) request with the specific ICCID has been removed(*terminal_iccid*) to the ECS together with the end user input.
14. The SP's back-end system managing subscription receives the subscription recovery request from the ECS one the end user input validation pass.
15. A set of eSIM profile requests over the ES2+ interface (for example, DownloadOrder, ConfirmOrder and ReleaseProfile) is made to the SM-DP+, for the new eSIM profile associated with the device eSIM, resulting in an activation code and ICCID for the primary device.
16. The SP back-end provides the ECS with an activation code or new ICCID and SM-DP+ address.
17. The ECS generates a 200 OK response with a PrimaryDeviceConfiguration entry for the recovered subscription bearing the ACTIVATED status (value of 1) and a filled in DownloadInfo structure.
18. The Primary ODSA device application indicates the eSIM to download the new eSIM profile.
19. The device's eSIM gets the eSIM profile from the SM-DP+ via ES9+ channel.
20. Optional - The Primary ODSA app makes another **ManageSubscription** to the ECS to provide/confirm the download of the newly created ICCID and to validate that the primary device subscription is ready and in proper activated state.
21. The ECS queries the Subscription Management system.
22. The ECS generates the proper response with subscription result (3-DONE).
23. Optional - The Primary ODSA client application makes an **AcquireConfiguration** request to the ECS to verify that the subscription and service for the device are in the proper states.
24. The ECS queries the SP's back-end system managing the subscriptions and profiles.
25. The ECS processes the response from the SP's back-end system and generates the proper 200 OK response containing a PrimaryConfiguration entry for the newly recovered subscription bearing the ACTIVATED status (value of 1).
26. As the primary device’s subscription and service is in the right state, the primary device can initiate cellular service.

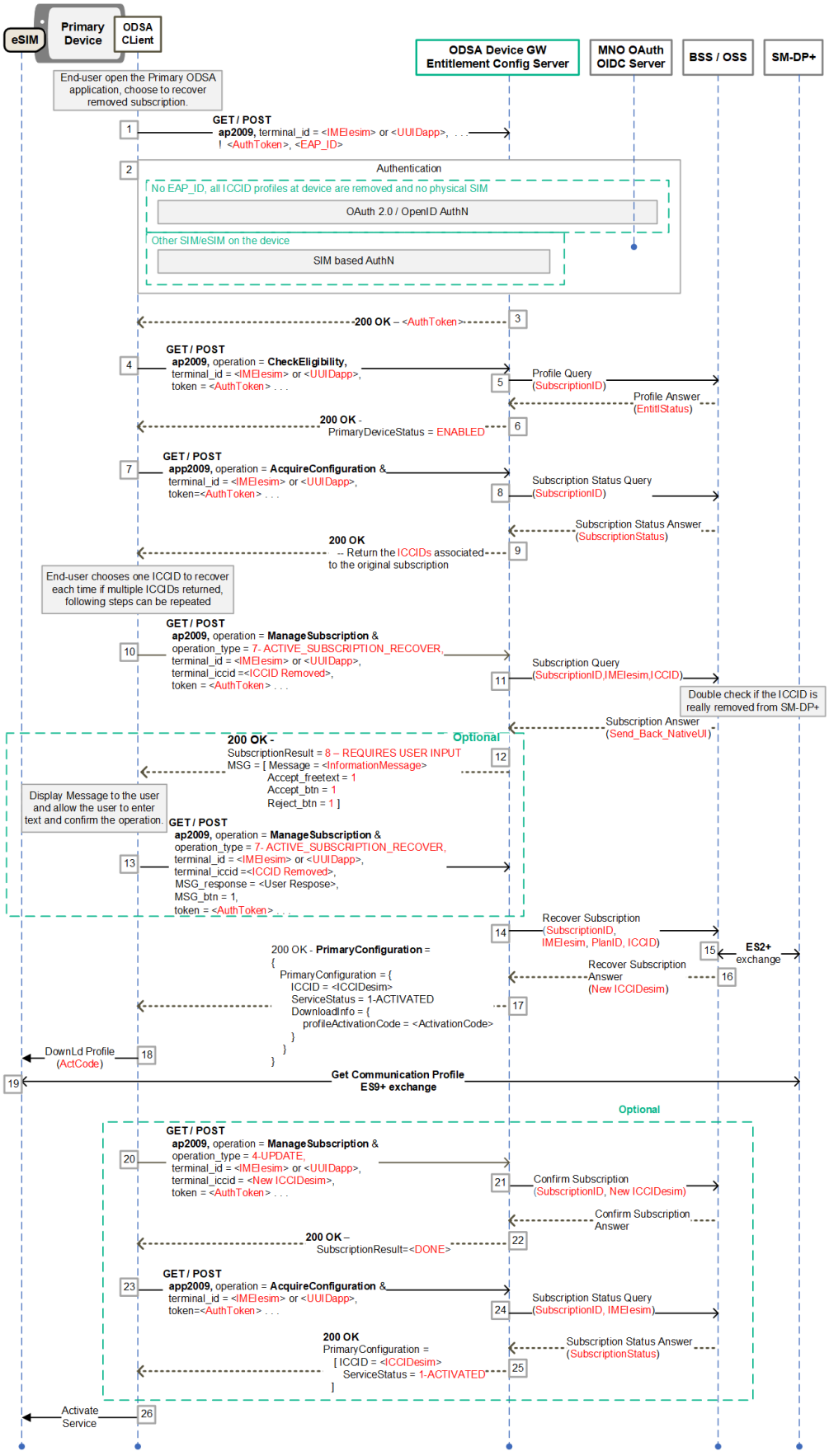

Figure 47. End user request for recovering the subscription which has been removed via native UI.

9 Data Plan Related Information Entitlement Configuration

Mobile devices that support high data rate Radio Access Types (RAT) can receive guidance from the Service Provider on how certain data-intensive and low-latency applications should access the device's available RATs.

As opposed to device or application configuration that is applied to all devices by a Service Provider, the Data Plan Related Information described in this clause is based on the end-user's subscription and associated plans.

The Data Plan Related Information is relayed by the requesting device to the applications using a method outside the scope of this specification. The returned configuration data contains the type of data plans associated with the end-user's subscription and assigned (if available) to each device’s RAT.

This is especially relevant for devices with 5G access which offers high-speed, high-volume data connectivity to the device's applications. With the inappropriate data plan in place, applications could exceed the usage limits of the subscription's data plan and result in a negative user experience due to data overage fees.

The device must therefore be made aware of the types of data plans active on the current subscription (for each RAT if applicable) and current data usage of the subscription and provide that information to target applications that are data and bandwidth-intensive. The device's subscription is identified through the authentication feature of TS.43, preferably via the EAP-AKA method (see 2.8.1) as it is seamless for the end-user and involves in a secure manner the device's SIM.

In addition to RAT related information, Data Plan information can include data boost information related to the access to slicing resources of the 5G network.

NOTE: use cases on 5G network resources other than network slicing are for further study.

More specifically use cases may require a performance boost upsell to the end user may require an entitlement check for the purposes of validating a subscriber’s price plan or checking Network’s current ability to support such an upsell experience for the user. This is especially relevant for devices with 5G SA access that have the ability to offer high-speed, low-latency data connectivity to the device’s applications.

The device may relay to the network the type of contextual experience of interest to the user in real time by means of a boost type. The network may validate that request against subscriber’s eligibility and network’s current ability to deliver that experience. For example, device may request a gaming experience based on user’s engagement in a gaming app and the network may deliver the necessary policy required to enable a gaming package upsell to the user in response.

Validation of subscriber price plan may include whether an upsell should be precluded due to various reasons e.g., user being on a premium price plan that inherently allows such experiences, or user belonging to certain category such as enterprise etc.

Validation of Network's ability to deliver the upsell experience may include current availability of Network resources or Network functionality to deliver the experience. How the network determines its ability is outside the scope of this document.

Figure 48 presents the high-level architecture of the Data Plan Related Information use case.

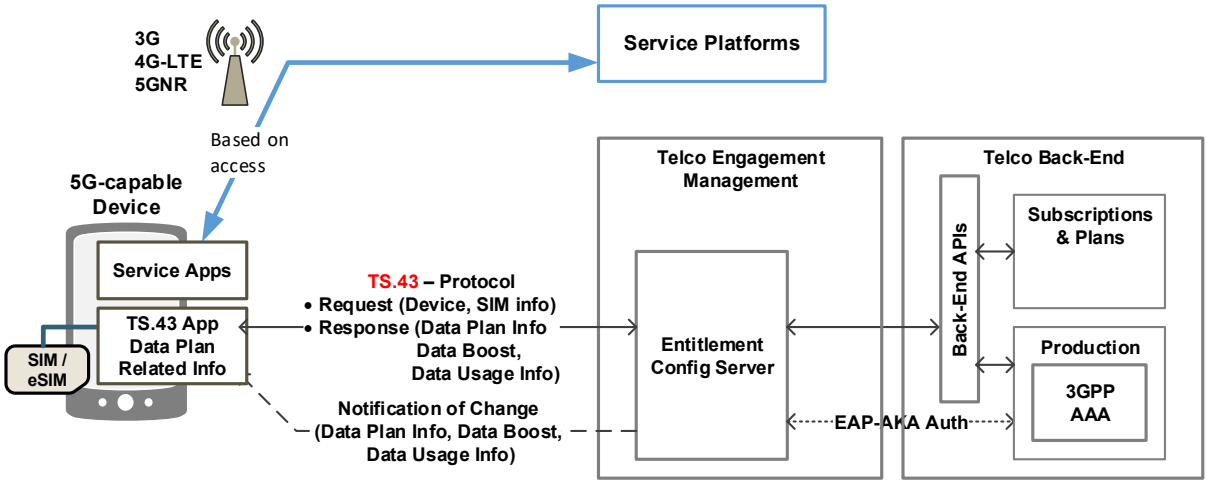

Figure 48. Data Plan Related Information high-level architecture

9.1 Data Plan Related Configuration Parameters

An ECS can implement either or all of the Data Plan, Data Boost or Data Usage Information function. The examples in this document show an ECS that implements both.

9.1.1 Data Plan Information Configuration Parameters

- Data Plan parameter names and presence:
  - `DataPlanInfo`: Top level, list of all data plan information associated with the device's subscription.
  - `DataPlanInfoDetails`: Within `DataPlanInfo`, one or more

`DataPlanInfoDetails` is a multi-parameter structures that provides data plan information for a particular Radio Access Types (RAT). The `DataPlanInfoDetails` structure has the parameters listed in Table 74.

"Data Plan Info" configuration parameters	Type	Values	Description
AccessType	Integer	0 to 5	The Radio Access Type (RAT) associated with the Data Plan
		0 - all	All the different RAT on the device
		1 – WiFi	Wi-Fi access type
		2 – 2G	RAT of type 2G
		3 – 3G	RAT of type 3G

"Data Plan Info" configuration parameters	Type	Values	Description
		4 - LTE	RAT of type LTE (4G)
		5 – NG-RAN	RAT of type NG-RAN (5G)
DataPlanType	String	Metered	The data plan is of the metered type
		Unmetered	The data plan is of the un-metered type

Table 74. Data Plan Information Configuration Parameter

9.1.2 Data Boost Information Configuration Parameters

- Data Boost parameter names and presence:
  - `DataBoostInfo`: Top level, list of all data plan slicing boost related information associated with the device's subscription.
  - `DataBoostInfoDetails`: Within `DataBoostInfo`, one or more

`DataBoostInfoDetails` is a multi-parameter structures that provides data plan information for a particular 5G slicing boost. The `DataBoostInfoDetails` structure has the parameters listed in Table 75.

"Data Boost Info" configuration parameters	Type	Values	Description
BoostType	Integer	0 - REALTIME_INTERACTIVE_TRAFFIC	Data Boost Type enabling users to consume to a real time interactive experience
		1 through 255 - The connection capability identifier defined in 3GPP TS24.526 Section 5.2 [20], which is encoded in one octet for the connection capability, is used as the value in Values of BoostType, e.g. 166 for Real time interactive.	The type of BoostType can be specified as connection capabilities defined in 3GPP TS24.526 Section 5.2 [20], e.g. Real time interactive.
BoostTypeStatus	Integer	0 - DISABLED	The Data Plan is eligible for this particular Boost Type; device should not offer notification and upsell experience but can poll later
		1 - ENABLED	The Data Plan is eligible to this particular Boost Type; device may offer notification and upsell experience

"Data Boost Info" configuration parameters	Type	Values	Description
		2 - INCOMPATIBLE	The Data Plan is not eligible for this particular Boost Type
TargetCharacteristics Info (Optional)	Structure	Multi-parameter value – see next table for detail	The values indicate target values of expected network performance for a corresponding BoostType

Table 75. Data Boost Information Configuration Parameters

Note: The value 0 in BoostType for REALTIME_INTERACTIVE_TRAFFIC is used for backword compatibility with TS.43 versions 11 and earlier. There is also a value for REALTIME_INTERACTIVE_TRAFFIC in TS24.526 [20]. The value is 166. These values, 0 and 166, are treated with same behaviour for REALTIME_INTERACTIVE_TRAFFIC.

The `TargetCharacteristicsInfo` configuration parameter is defined as a structure with several parameters as shown in Table 76

"TargetCharacteristicsInfo" configuration parameters	Type	Values	Description
PDB (Optional)	Integer	A valid positive integer number excluding 0 value.	The value indicates a packet delay budget which users can expect as network performance at the time of Data Boost. Unit is ms.
Jitter (Optional)	Integer	A valid positive integer number excluding 0 value.	The value indicates a jitter which users can expect as network performance at the time of Data Boost. Unit is ns.
MinDownlinkDataRate (Optional)	Integer	A valid positive integer number including 0 value.	The value indicates a minimum downlink data rate which users can expect as network performance at the time of Data Boost. Unit is Mbps.
MaxDownlinkDataRate (Optional)	Integer	A valid positive integer number including 0 value.	The value indicates a maximum of downlink data rate. Unit is Mbps.
MaxDownlinkBurstRate (Optional)	Integer	A valid positive integer number including 0 value.	The value indicates a maximum downlink burst rate that will enable the network to burst data at a higher rate than the BoostedMaxDownlinkDataRate for a period of time. Unit is Mbps.

"TargetCharacteristicsInfo" configuration parameters	Type	Values	Description
MinUplinkDataRate (Optional)	Integer	A valid positive integer number including 0 value.	The value indicates a target minimum uplink data rate which users can expect as network performance at the time of Data Boost. Unit is Mbps.
MaxUplinkDataRate (Optional)	Integer	A valid positive integer number including 0 value.	The value indicates a maximum of uplink data rate. Unit is Mbps.
MaxUplinkBurstRate (Optional)	Integer	A valid positive integer number including 0 value.	The value indicates a maximum uplink burst rate that will enable the network to burst data at a higher rate than the BoostedMaxUplinkDataRate for a period of time. Unit is Mbps.
PER (Optional)	Integer	A valid positive integer number including 0 value.	The value indicates a packet error rate which users can expect as network performance at the time of Data Boost. The value specifies the x of "10^-x"

Table 76. TargetCharacteristicInfo Configuration Parameters

The usage of the following parameters is noted that:

- **`MinDownlinkDataRate`** and **`MinUplinkDataRate`** show lower bound of data rate to be provided by a network associated with a boost type. For example, the parameters are specified for boost types which always require to consume a certain data rate, such as streaming services.
- **`MaxDownlinkDataRate`** and **`MaxUplinkDataRate`** show upper bound of data rate to be provided by a network associated with a boost type. For example, the parameters are specified for IoT-related boost types. They enable operators to save radio resource consumption, which leads to provide reasonable services to customers.

9.1.3 Data Usage Information Configuration Parameters

- Data Usage parameter names and presence:
  - `DataUsageInfo`: Top level, list of all data usage information associated with the device's subscription.
  - `DataUsageInfoDetails`: Within `DataUsageInfo`, one or more

`DataUsageInfoDetails` is a multi-parameter structures that provides information on current data usage over cellular. The `DataUsageInfoDetails` structure has the parameters listed in Table 77.

"Data Usage Info" configuration parameters	Type	Values	Description
DataUsageType	Integer	0 to 1	The type of data usage
		0 - Cellular	Cellular data for this device
		1 - Tethering	Cellular data to connect other device(s) to the cellular network via this device (e.g. mobile hotspot, USB tethering)
DataUsageName (Optional)	String	Any string value	Name of the data usage provided by the MNO
DataUsageDescription (Optional)	String	Any string value	Description of the plan offered by the MNO. It is considered as an optional parameter, but it is recommended to convey additional information.
EndOfBillingCycle (Conditional)	Timestamp	ISO 8601 format, of the form YYYY-MM-DDThh:mm:ssTZD	This UTC value provides the expiration time for current billing cycle. This parameter shall not be present if there is no expiration time for current billing cycle.
DataAllowanceInBytes (Conditional)	Integer	A valid positive integer number including 0 value	Indicates the data allowance for the current billing cycle in bytes. This parameter shall not be present if the data allowance is unlimited.
DataUsedInBytes	Integer	A valid positive integer number including 0 value	Indicates the used data for the current billing cycle in bytes.
StreamingApplicationMaxDownlinkDataRate (Optional)	Integer	A valid positive integer number including a 0 value	Indicates the maximum downlink rate in Kbps to configure streaming applications.

Table 77. Data Usage Information Configuration Parameters

9.1.4 5G SA Information Configuration Parameters

- 5G Standalone (SA) Information parameter names and presence:
  - `5GSAInfo`: Top level, list 5G SA information associated with the device's subscription.
  - `5GSAInfoDetails`: Within `5GSAInfo`

`5GSAInfoDetails` is a multi-parameter structures that provides information on users 5G-SA enablement by the network. The `5GSAInfoDetails` structure has the parameters listed in Table 78.

"5GSAInfo" configuration parameters	Type	Values	Description
5GSAStatus	Integer	0 - DISABLED	5G-SA disabled for this device
		1 - ENABLED	5G-SA enabled for this device

Table 78. 5G SA Information Configuration Parameters

9.2 Data Plan Related Information Response Example

Table 79 presents an example for a returned Data Plan Related Information entitlement configuration in XML format where the only RAT that is metered is NG-RAN (5G).

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS">
         <parm name="version" value="1"/>
         <parm name="validity" value="172800"/>
     </characteristic>

     <characteristic type="TOKEN">
         <parm name="token" value="ASH127AHHA88SF"/>
     </characteristic>

     <characteristic type="APPLICATION">
        <parm name="AppID" value="ap2010"/>
        <characteristic type="DataPlanInfo">

            <characteristic type="DataPlanInfoDetails">
                <parm name="AccessType" value="1"/>
                <parm name="DataPlanType" value="Unmetered"/>
            </characteristic>

            <characteristic type="DataPlanInfoDetails">
                <parm name="AccessType" value="2"/>
                <parm name="DataPlanType" value="Unmetered"/>
            </characteristic>

            <characteristic type="DataPlanInfoDetails">
                <parm name="AccessType" value="3"/>
                <parm name="DataPlanType" value="Unmetered"/>
            </characteristic>

            <characteristic type="DataPlanInfoDetails">
                <parm name="AccessType" value="4"/>
                <parm name="DataPlanType" value="Unmetered"/>
            </characteristic>

            <characteristic type="DataPlanInfoDetails">
                <parm name="AccessType" value="5"/>
                <parm name="DataPlanType" value="Metered"/>
            </characteristic>

        </characteristic>

       <characteristic type="DataBoostInfo">
            <characteristic type="DataBoostInfoDetails">
                /* REALTIME_INTERACTIVE_TRAFFIC */
                <parm name="BoostType" value="166"/>
                <parm name="BoostTypeStatus" value="1"/>
                <characteristic type="TargetCharacteristicsInfo">
                    <parm name="PDB" value="20"/>
                    <parm name="PER" value="3"/>
            </characteristic>
        </characteristic>

        <characteristic type="DataUsageInfo">

            <characteristic type="DataUsageInfoDetails">
                <parm name="DataUsageType" value="0"/>
                <parm name="DataUsageName" value="Unlimited Data"/>
                <parm name="DataUsageDescription" value="This is the description
of the Unlimited Data"/>
                <parm name="EndOfBillingCycle" value="2023-02-28T23:59:99"/>
                <parm name="DataUsedInBytes" value="2147483648"/>
            </characteristic>

            <characteristic type="DataUsageInfoDetails">
                <parm name="DataUsageType" value="1"/>
                <parm name="DataUsageName" value="Tethering data up to 5 GB"/>
```

```
                <parm name="DataUsageDescription" value="This is the description
of the Tethering data up to 5 GB"/>
                <parm name="EndOfBillingCycle" value="2023-02-28T23:59:99"/>
                <parm name="DataAllowanceInBytes" value="5368709120"/>
                <parm name="DataUsedInBytes" value="314572800"/>
            </characteristic>

        </charateristic>

     </characteristic>
</wap-provisioningdoc>
```

Table 79. Example of a Data Plan Related Information response in XML format

Table 80 presents an example for a returned Data Plan Related Information entitlement configuration in JSON format where only 3G, LTE and NG-RAN data plan info details are returned, and both LTE and NG-RAN are metered.

```
{
    "Vers" : {
        "version" : "1",
        "validity" : "172800"
    },
    "Token" : {              // Optional
        "token" : "ASH127AHHA88SF"
    },
    "ap2010" : {            // Data Plan Information app
        "DataPlanInfo" : [{
            "DataPlanInfoDetails" : {
                "AccessType" : "3",
                "DataPlanType" : "Unmetered" }
            },
            {
            "DataPlanInfoDetails" : {
                "AccessType" : "4",
                "DataPlanType" : "Metered" }
            },
            {
            "DataPlanInfoDetails" : {
                "AccessType" : "5",
                "DataPlanType" : "Metered" }
            }],

        "DataBoostInfo" : [{
            "DataBoostInfoDetails" : {
                 /* REALTIME_INTERACTIVE_TRAFFIC */
                "BoostType" : "166",
                "BoostTypeStatus" : "1"
                    "TargetCharacteristicsInfo" : {
                    "PDB" : "20",
                    "PER" : "3" }
                }
        }]
        "DataUsageInfo" : [{
            "DataUsageInfoDetails" : {
                 "DataUsageType" : "0",
                 "DataUsageName" : "Unlimited Data",
                 "DataUsageDescription" : "This is the description of the
Unlimited Data",
                 "EndOfBillingCycle" : "2023-02-28T23:59:99",
                 "DataUsedInBytes" : "2147483648"
            }
        },
        {
            "DataUsageInfoDetails" : {
                "DataUsageType" : "1",
                "DataUsageName" : "Tethering data up to 5 GB",
                "DataUsageDescription" : "This is the description of the
Tethering data up to 5 GB",
                "EndOfBillingCycle" : "2023-02-28T23:59:99",
                "DataAllowanceInBytes" : "5368709120",
                "DataUsedInBytes" : "314572800"
            }
        }
      ]
    }
}
```

Table 80. Example of a Data Plan Related Information response in JSON format

9.3 Data Plan Related Information Call Flow

Figure 49 shows the call flow for the Data Plan Related Information entitlement configuration use case. Authentication steps are not shown for simplification purposes.

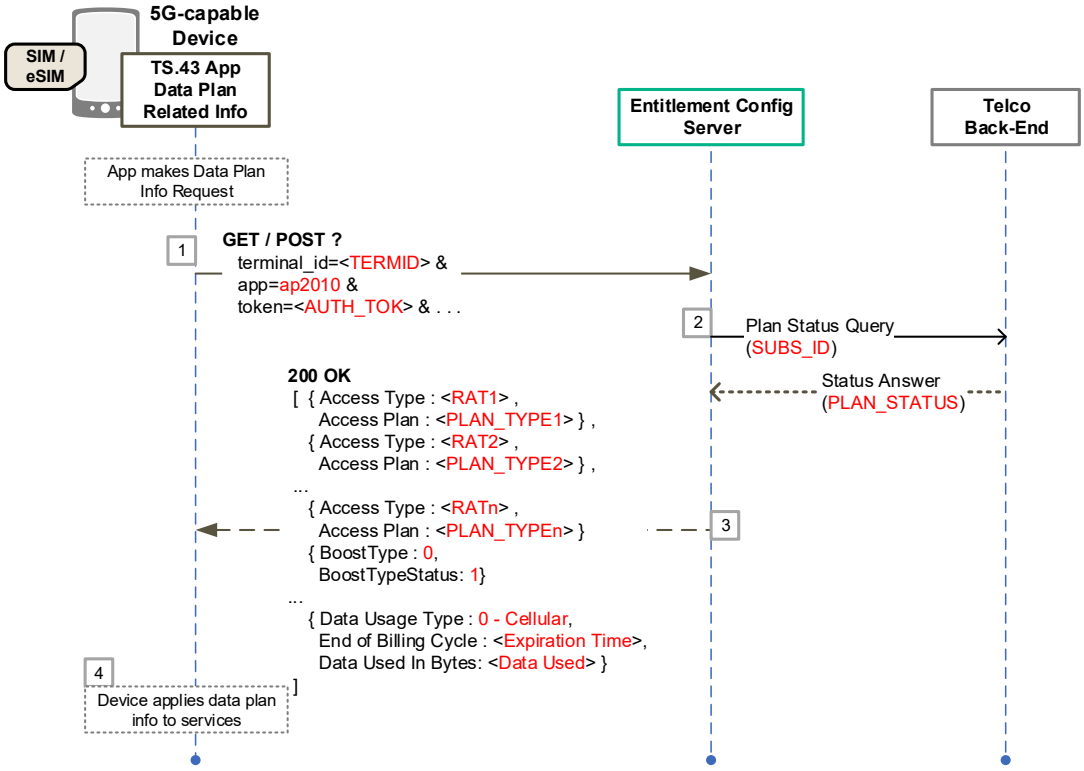

Figure 49. Data Plan Related Information Call Flow

The steps are:

1. The device makes a Data Plan Related Information entitlement request with proper App ID and token acquired from an authentication exchange.
2. The ECS queries the Service Provider's back-end system for data plan related information associated with the end-user's subscription.
3. The ECS receives the data plan related information and creates an entitlement response of the proper format.
4. The device applies the data plan and/or boost info details and/or data usage info details for the targeted application(s).

If there is some change in plan status that could impact on the data plan related information, the ‘Telco Back-End’ will inform the ECS about this change. ECS will notify to the device using any of the available options (see section 2.6) to refresh this data as shown in the Figure 50.

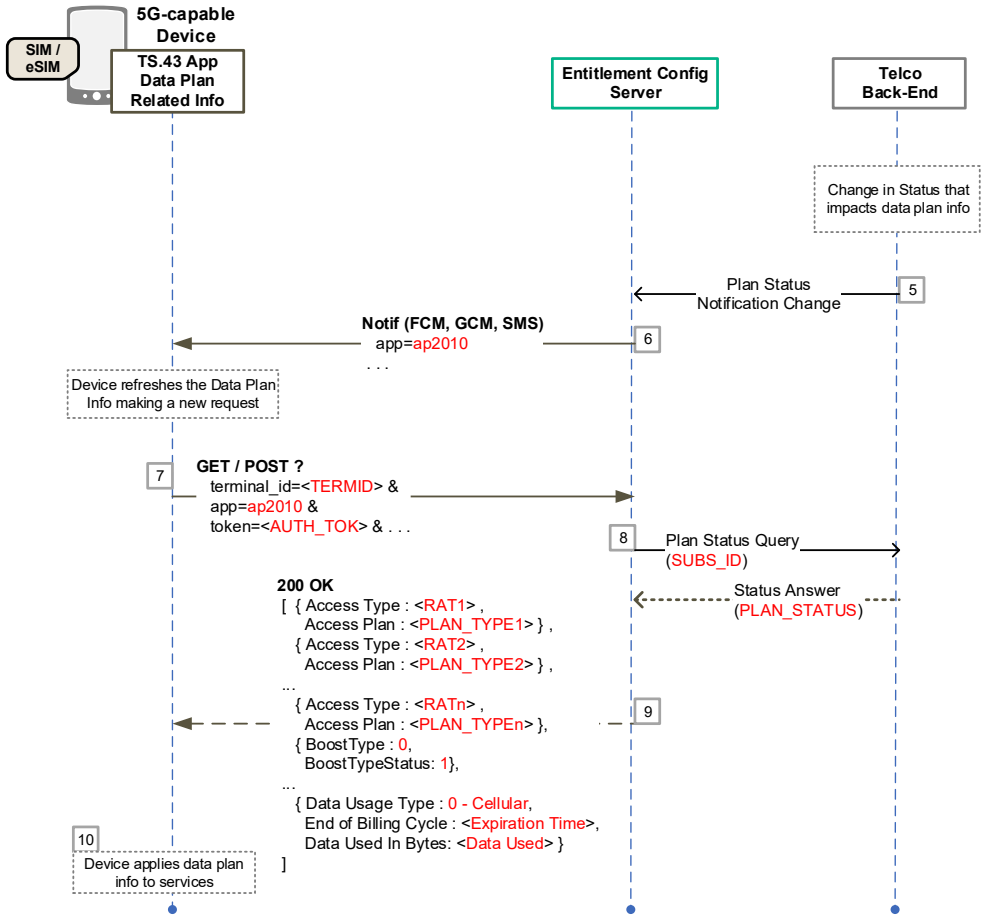

Figure 50. Data Plan Related Information request triggered by carrier notification.

The steps are:

5. Service Provider informs the ECS of a change in data plan related information.
6. The ECS generates the notification message based on the notify_* parameters received earlier from the device (see 2.6 for details). This notification will trigger a new Data Plan Information entitlement request as detailed in Figure 50.
7. **Steps 7 to 10** are exactly the same as steps 1 to 4 detailed in Figure 49.

9.4 Data Boost real-time request

The device configured with a particular Data Boost Type can request this Boost Type to the ECS and the ECS can initiate the webview procedures in order to complete the transaction.

As the state of the data boost provisioning and its eligibility on the network can be very fluid. The device also can receive critical data boost status information in real-time to provide best user experience.

9.5 Data Boost Web View Parameters

These are the parameters name and presence required in Data Boost.

- `ServiceFlow_URL`: Conditional
- `ServiceFlow_UserData`: Conditional

- `ServiceFlow_ContentsType`: Conditional

During the activation of Data Boost, end-users can be presented with web views specific to the carrier. Data boost web views allow end-users to change user-specific attributes of Data Boost, like the acceptance of the service's Terms and Conditions (T&C) or purchasing a Data Boost.

The entitlement parameters associated with Data Boost are described in Table 81.

Data Boost Entitlement parameter	Type	Values	Description
ServiceFlow_URL (Conditional)	String	URL to a Service Provider site or portal	The URL of web views to be used by Data Boost client to present the user with Data Boost service management, which may include agreeing to the T&C of the Data Boost service or purchasing a Data Boost.
ServiceFlow_UserData (Conditional)	String	Parameters or content to insert when invoking URL provided in the `ServiceFlow_URL` parameter	User data sent to the Service Provider when requesting the `ServiceFlow_URL` web view. It should contain user-specific attributes to improve user experience. The format must follow the `ServiceFlow_ContentsType` parameter. For content types of JSON and XML, it is possible to provide the base64 encoding of the value by preceding it with `encodedValue=`.
ServiceFlow_ContentsType (Conditional)	String	Specifies content and HTTP method to use when reaching out to the web server specified in `ServiceFlow_URL`.	
		NOT present	Method to `ServiceFlow_URL` is HTTP GET request with query parameters from `ServiceFlow_UserData`.
		json	Method to `ServiceFlow_URL` is HTTP POST request with JSON content from `ServiceFlow_UserData`.
		Xml	Method to `ServiceFlow_URL` is HTTP POST request with XML content from `ServiceFlow_UserData`.

Table 81. Data Boost Service Parameters - WebView Information

9.6 Data Boost Web View JavaScript Callbacks

At the completion of the web service flow, the web service shall invoke a specific JavaScript (JS) callback function associated with the Data Boost manager. The callback functions shall provide the overall state of the web flow to the Data Boost manager and indicate that the webview needs to be closed.

The object associated with the callback functions is **DataBoostWebServiceFlow** and three different callback functions are defined to reflect the state of the web logic.

9.6.1 notifyPurchaseSuccessful(duration)

Calling this method indicates that the user has successfully purchased data boost.

The parameter `duration` is mandatory. It is the time period (in milliseconds) for which the boost is applied.

After this call back is called, the webview is closed.

9.6.2 notifyPurchaseFailed(code, reason)

Calling this method indicates that the data boost purchase has failed.

The parameter `code` is mandatory. The parameter `reason` is optional. Details for these parameters are provided in Table 82.

After this call back is called, the webview is closed.

	Type	Values	Description
code	Integer	0 – FAILURE_CODE_UNKNOWN	Unknown failure code (in this case the parameter `reason` provides a human-readable reason)
		1 – FAILURE_CODE_AUTHENTICATION_FAILED	User authentication failed
		2 - FAILURE_CODE_PAYMENT_FAILED	User payment failed
reason	String	ANY VALUE	Human readable reason for the failure.

Table 82. Failure codes for data boost purchase failure

9.6.3 dismissFlow()

Calling this method indicates that the data boost purchase mechanism has ended prematurely, either caused by user action or by an error in the web sheet logic or from the network side.

After this call back is called, the webview is closed.

9.7 Data Boost Real-time Request Parameters

- Parameter names and presence:
  - `boost type`: Top level; list of performance experience in the form of a boost type category. See Table 83 for currently defined values for this version.

"Data Boost Real-time" configuration parameters	Type	Values	Description
boost_type	Integer	See `BoostType` in Table 75	Boost type to be requested by the subscriber

Table 83. Data Boost Real-time Request Parameter

9.8 Data Boost Real-time Request Example

Table 84 presents an example for the Data Boost operation for a server ODSA application.

```
GET ? terminal_id = 013787006099944&
token = es7w1erXjh%2FEC%2FP8BV44SBmVipg&
terminal_vendor = TVENDOR&
terminal_model = TMODEL&
terminal_sw_version = TSWVERS&
entitlement_version = ENTVERS&
app = ap2010&
boost_type = 0&
vers = 1 HTTP/1.1

Host: entitlement.telco.net:9014
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Table 84. Example of a Data Boost request

9.9 Data Boost Real-Time Response Parameters

- Data Boost Real-time response parameter names and presence:
  - `EntitlementStatus`: provides the real-time entitlement status of the boost request by the device. See Table 85 for details.
  - `ProvStatus`: provides the real-time provisioning status of the boost request by the device. See Table 86 for details.

The real-time response includes an Entitlement status and Provisioning status as defined in the tables below. If a data plan is eligible for a boost experience, device may handle user interaction based on internal logic (outside the scope of this spec). The entitlement response may also provide a redirect URL from where the user is able to purchase the boost experience.

	Type	Values	Description
EntitlementStatus	Integer	0 - DISABLED	Data Plan is eligible, but boost is disabled currently; device should not offer notification and upsell experience but can poll later
		1 – ENABLED	Data Plan is eligible. Boost is allowed, provisioned, and activated; device may offer notification and upsell experience
		2 – INCOMPATIBLE	Data Plan is no longer eligible. Boost is not allowed or can't be offered; device should not offer upsell experience
		3 - PROVISIONING	Data Plan is eligible. Boost is not fully provisioned; device should wait for provisioning to finish
		4 - INCLUDED	Data Plan is eligible. Boost is enabled e.g. included in the sub plan. Device may proceed with upsell experience, but notification is not required

Table 85. Real-time Data Boost Information Configuration Parameter

The Provisioning status provides the device with additional real-time information regarding the provisioning status of the boost service. If the provisioning is pending, the device may implement logic to delay the boost purchase.

	Type	Values	Description
ProvStatus	Integer	`0 – NOT PROVISONED`	Boost service is not provisioned yet on the backend
		`1 – PROVISIONED`	Boost service is fully provisioned on the backend
		`2 – NOT AVAILABLE`	Boost service provisioning progress not required/tracked
		`3 – IN PROGRESS`	Boost service provisioning is still in progress; client should wait for provisioning to complete.

Table 86. Provisioning status Information Configuration Parameter

9.10 Data Boost Real-time Response Example

Table 87 presents an example for a returned Data Boost Real-time Information entitlement configuration in XML format.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS">
         <parm name="version" value="1"/>
         <parm name="validity" value="172800"/>
     </characteristic>

     <characteristic type="TOKEN">
         <parm name="token" value="ASH127AHHA88SF"/>
     </characteristic>

     <characteristic type="APPLICATION">
        <parm name="AppID" value="ap2010"/>
        <parm name=”EntitlementStatus” value=”1”/>
        <parm name=”ServiceFlow_URL” value=”X”/>
        <parm name=”ServiceFlow_UserData” value=”X”/>
        <parm name=”ProvStatus” value=”1”/>
     </characteristic>
</wap-provisioningdoc>
```

Table 87. Example of a Data Boost Real-time Information response in XML format

9.11 Data Boost Real-time Request Call Flow with webview

Figure 51 shows the call flow for the Data Boost Upsell Information entitlement configuration use case. Authentication steps are not shown for simplification purposes.

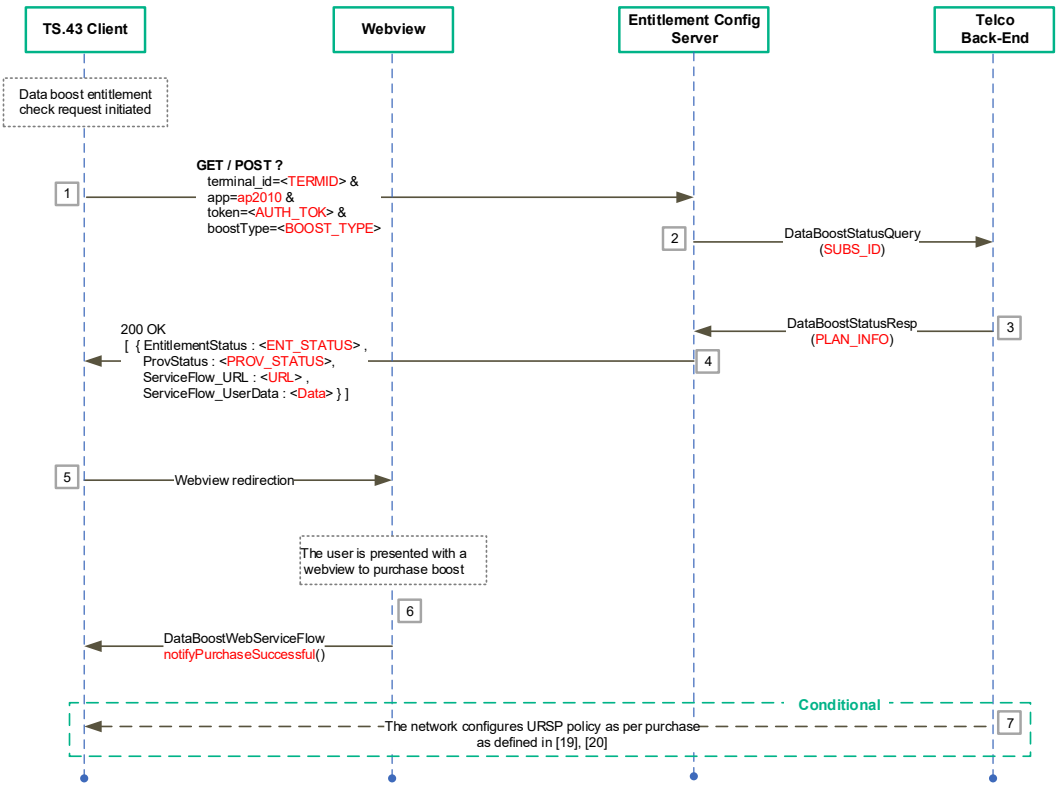

Figure 51. Data Boost Real-time request and response Call Flow with webview

The steps are:

1. Once a data boost entitlement check request is initiated, the device entitlement client makes a Performance Boost Upsell Information entitlement request with proper App

ID, optional OS App ID and token acquired from an authentication exchange. The entitlement client also provides the boost type corresponding to the upsell experience requested by the user.

2. The ECS queries the Service Provider's back-end system for plan information associated with the end-user's subscription.
3. The ECS receives the plan information and Network capability information from the Service Provider’s back-end system.
4. The ECS creates an entitlement response (data boost real-time response) of the proper format and informs the device entitlement client.
5. The device entitlement client informs the user of availability of data boost experience. If the user requests the data boost experience, the user is redirected to a webview to purchase the data boost experience. Aspects related to user consent for notification and details of when and how the notification of data boost purchase availability is performed is outside the scope of this specification.
6. The user is presented with the webview to purchase the boost and the webview invokes a callback function to inform the device entitlement client of data boost purchase decision.
7. Conditional (If not received already or expired): Depending on the data boost experience purchased by the user, the Service Provider’s back-end configures the device with the appropriate URSP policy as specified in [19] and [20].

10 Server-initiated ODSA Procedure Call Flows

In specific environments like the enterprise one, there are some needs to manage the device subscriptions. This could be managed by Mobile Device Management (MDM) software for the purpose to simplify and enhance the management of the end user devices.

The activation flow for the new devices is similar to the one implemented for the Companion devices (see section 7) where the MDM works as a primary device and the end user device as a companion one.

One of the main differences is that behind the device (MDM system) initiating the request there is no user, neither an eSIM/SIM but just a server. Due to this restriction, it is no possible to use authentication methods like Embedded EAP-AKA (see section 2.8.1) or OAuth2.0/OpenID with customer interaction (see section 2.8.2) and it is necessary to use Server to Server Authentication using OAuth2.0 as described in section 2.8.3 of this document.

The architecture for the server-initiated ODSA use case is shown in Figure 52. The Entitlement Configuration Server acts as the Service Provider's ODSA Gateway for the ODSA procedure (labelled as the "ODSA GW" in Figure 52), providing entitlement and configuration data to the server (MDM) managing the devices for "ODSA server-initiated" application.

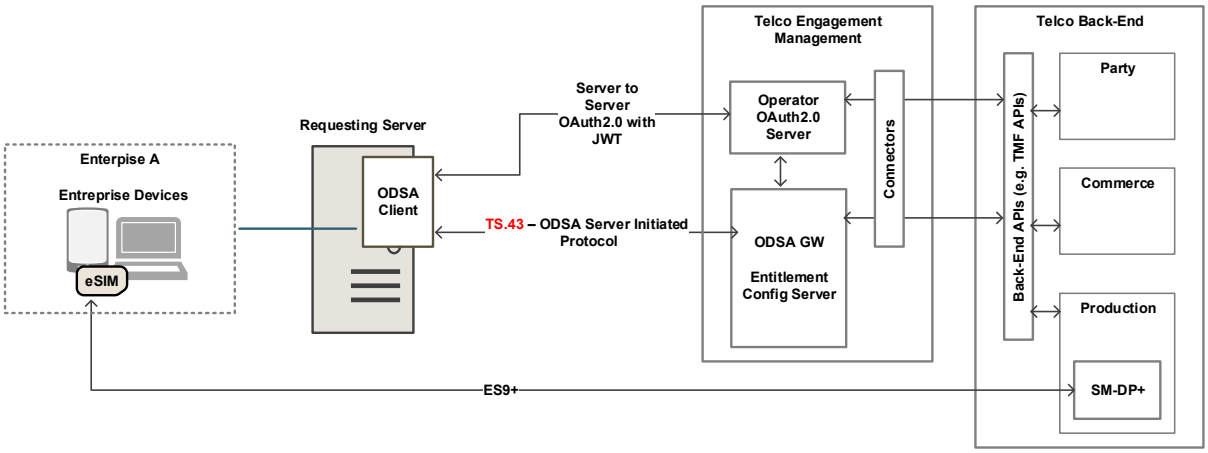

Figure 52. ODSA server-initiated request, architecture, and TS.43 positioning

10.1 Initial considerations

The main difference between this use case and the others related to ODSA is that there is not any direct interaction with the use, and the device doesn't interact with the entitlement configuration server (ECS) until it is already activated. At that point on time, and if the policies applied by the MDM allow it to do that, it could interact as any other device having the proper TS.43 apps.

The MDM is not a terminal but a server, but, even so, in the request there will be some parameters referring to terminal_* present on the requests as part of the RCC.14 standard. For these mandatory parameters, it is recommended to use dummy values, keeping the new ones (`requestor_id` or `enterprise_*` as referred in Table 27).

Due to there is no real info for the targeted device in the `CheckEligibility` request, it should be the MDM the one in charge of checking the eligibility of the device to use any specific service when onboarding with a new plan. These policies/rules are managed by the MDM and are out of the scope of this spec.

10.2 Subscription Activation initiated by the server.

The following premises are considered for this the case:

- The requesting server (through the ODSA client application) is allowed to request new eSIM profiles for and specific Enterprise (`enterprise_id`).
- The ODSA GW (Entitlement Configuration Server) is able to keep the authentication tokens for each requesting server (`requestor_id`) and enterprise (`enterprise_id`) to avoid sending the `enterprise_id` in each request triggered by the requesting server once it has the authentication token.
- If the authentication token is invalid or expires, the server initiating the ODSA request will need to get a new Access Token (from the Authorization server) to perform the new Authentication through the ECS (Resource Server).

Figure 53 shows the steps of the flow for the activation of an eSIM managed by the requesting server (aka MDM).

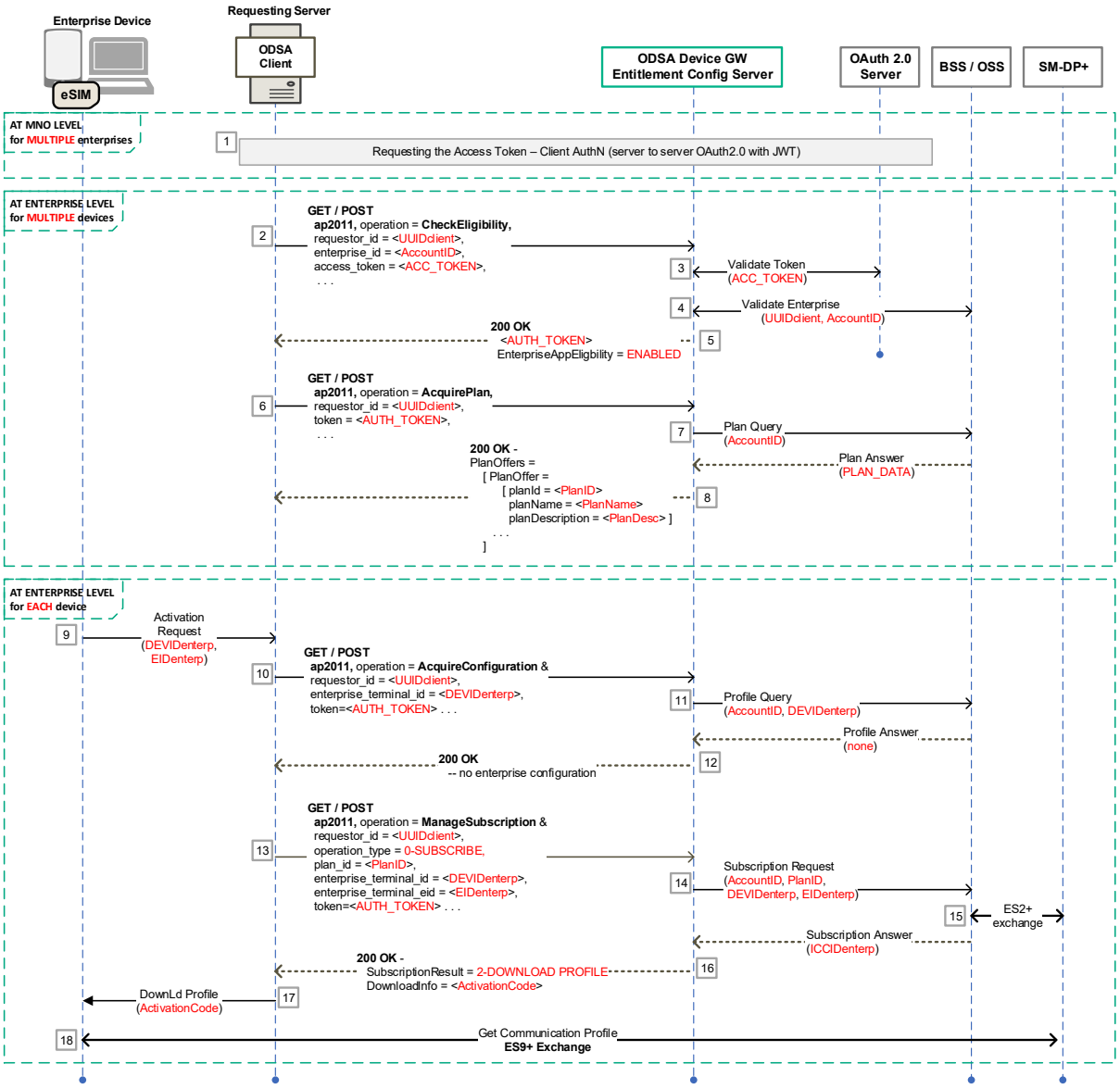

Figure 53. ODSA initiated by a server flow.

The steps are the following ones and can be split in three sections:

Steps at MNO level for MULTIPLE enterprises:

1. The server ODSA application requests (and gets) an access token to the SP's Authentication Server. For additional info about how the requesting server gets the access token see section 2.8.3.

Steps at enterprise level for MULTIPLE devices:

2. The server ODSA application makes a **CheckEligibility** request to the ECS providing the access token (`ACC_TOKEN`) and the Enterprise ID (`enterprise_id`) to operate.
3. The ECS validates the access token with SP OAuth2.0 Server.

4. Additional to the access token validation, the ECS checks if Enterprise is entitled to manage subscriptions.
5. Once access token validation and enterprise entitlement check are successful, the ECS will create an AuthN Token that will be sent back to the ODSA client application. The ECS will associate this token to the ODSA app ID (`requestor_id`) and Enterprise ID for future requests. This avoids sending the Enterprise ID in each request.
6. The server ODSA application makes an **AcquirePlan** request to get all the plans offered by the SP to a specific Enterprise. Note that it is not necessary to send the `enterprise_id` parameter as the ECS knows it based on the authentication token received.
7. The ECS queries, based on the `enterprise_id`, for this plan info to the SP back-end system managing this info.
8. The ECS generates a proper response with the different plans available for offering.

Steps at enterprise level for EACH device:

9. A new device (belonging to an enterprise) sends an activation request to the requesting server. This new device will be managed as an enterprise device for the requesting server.
10. The server ODSA client application makes an **AcquireConfiguration** request to the ECS to obtain information on any communication profiles associated with the device.
11. The ECS queries the SP's back-end system managing the subscriptions and active profiles.
12. The ECS processes the response from the SP's back-end system and generates the proper 200 OK response containing `EnterpriseDeviceConfigurations` without any `EnterpriseConfiguration` (no profile/subscription is associated with the enterprise device).
13. The server ODSA client application makes a **ManageSubscription** request to the ECS with an `operation_type` set to SUBSCRIBE (value of 0) to initiate the subscription procedure for the enterprise device.
14. The ECS makes a request towards the SP's back-end system to activate the selected plan and subscription.
15. The SP's back-end system interacts with the SM-DP+ over the ES2+ interface to make the required eSIM profile requests associated with the new subscription (for example, `DownloadOrder`, `ConfirmOrder` and `ReleaseProfile`) resulting in an activation code and ICCID for the enterprise device.
16. The ECS processes the response from the SP's back-end system and generates the proper **ManageSubscription** 200 OK response with a `SubscriptionResult` set to DOWNLOAD_PROFILE (value of 2), and a filled in `DownloadInfo` structure with the proper `ActivationCode`.
17. The server ODSA client application informs the enterprise device to download the eSIM profile.
18. The new device (acting as an enterprise one) downloads the eSIM profile from the SM-DP+.

10.2.1 Subscription Activation for Delayed Activations

It is possible that carrier could consider delaying the eSIM profile activation in their backend systems, so a polling or notification mechanisms should be implemented to notify when the eSIM profile is ready to be used.

In case of implementing the polling mechanism, it should be necessary to include the loop for refreshing status between steps 14 and 16 in the Figure 53 as explained in the section 7.3.

In case of implementing the notifications, and due to there is no standard notification API for these MDMs, carriers, ECS vendors and MDM vendors should agree the way to implement this. This specification/agreement is out of scope of TS.43.

11 Direct Carrier Billing Entitlement Configuration

The following sections describe the different configuration parameters associated with the Direct Carrier Billing (aka DCB) entitlement as well as the expected behaviour of the DCB client based on the entitlement configuration document received by the client.

Figure 54 shows the steps of the flow for the activation of DCB.

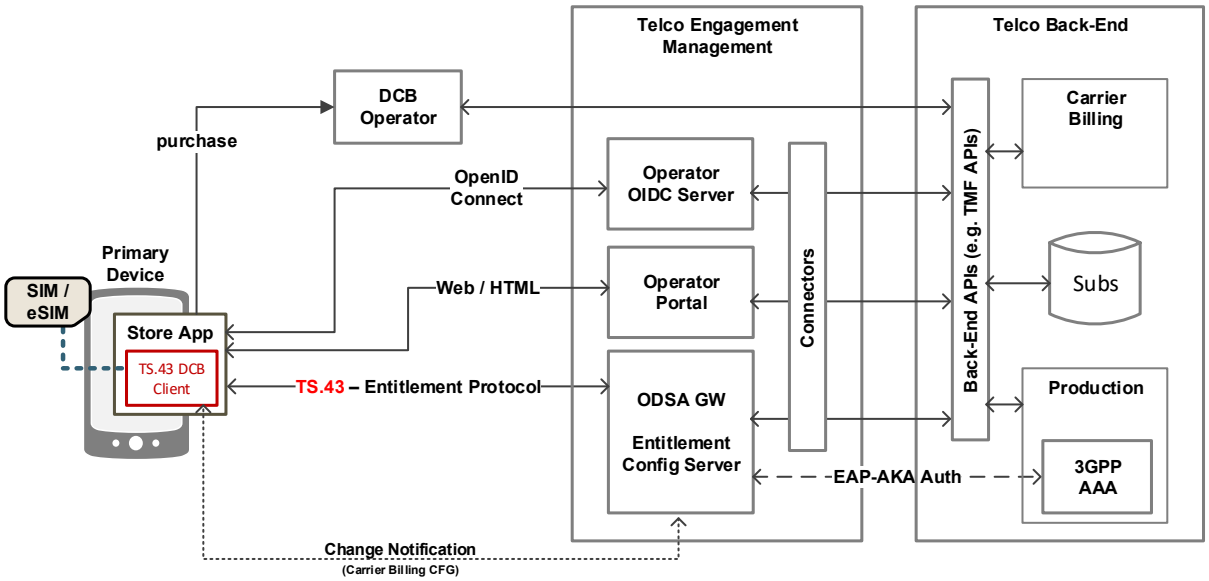

Figure 54. Direct Carrier Billing Configuration - High level Architecture

11.1 DCB Entitlement Parameters

Parameters for the DCB entitlement provide the overall status of the DCB service to the client, as well as the different sub-status associated with the activation procedure of the service.

The DCB entitlement parameters also include information associated with the web views presented to users by the DCB client during management of the service.

Additional to the parameters identified in (section 2.3, Table 4), new parameters are required for the Direct Carrier Billing use case. These parameters are defined in the following table:

HTTP GET parameter	Type	Description	Usage
store_vendor	String	This value shall be a unique and persistent identifier of the store. Example: S9999	Only required for DCB

HTTP GET parameter	Type	Description	Usage
store_user_id	String	User Identity on the store. This value shall be a unique and persistent identifier for each specific user in each specific store. This value is used to be generated by a system in the store.	Only required for DCB
MSG_btn (Optional)	Integer	This indicate either “Accept” or “Reject” button has been pressed on device UI. The action associated with is to set `TC_Status`.	
		0 – REJECTED	T&C have been rejected by the end-user. `TC_Status` will be set to 0 - NOT AVAILABLE
		1 – ACCEPTED	T&C have been accepted by the end-user. `TC_Status` will be set to 1 – AVAILABLE

Table 88. Additional GET Parameters for DCB Entitlement Configuration Request

11.1.1 DCB Entitlement Status

This is the parameter name and presence required in DCB.

- `EntitlementStatus`: Mandatory

This parameter indicates the overall status of the DCB entitlement, stating if the service can be offered on the device, and if it can be activated or not by the end-user.

The different values for the DCB entitlement status are provided in Table 89

DCB Entitlement parameter	Type	Values	Description
EntitlementStatus (Mandatory)	Integer	0 - DISABLED	DCB service allowed, but not yet provisioned and activated on the network side
		1 - ENABLED	DCB service allowed, provisioned, and activated on the network side
		2 - INCOMPATIBLE	DCB service cannot be offered
		3 - PROVISIONING	DCB service being provisioned on the network side

Table 89. Entitlement Parameter - DCB Overall Status

11.1.2 DCB T&C Status

These are the parameters name and presence required in DCB for T&C status.

- `TC_Status`: Mandatory

- `TC_Operation`: Optional

In some regions, end-users must agree to the Terms and Conditions (T&C) of the DCB service before being allowed to use it. This entitlement parameter indicates if that condition must be met before offering the DCB service.

Also, if acceptance of the DCB's T&C is indeed needed from the end-user, this parameter indicates the state of the "T&C acceptance" process.

The different values for the DCB T&C status are provided in Table 90.

DCB Entitlement parameter	Type	Values	Description
TC_Status (Mandatory)	Integer	0 - NOT AVAILABLE	T&C have not yet been accepted by the end-user
		1 - AVAILABLE	T&C have been accepted by the end-user
		2 - NOT REQUIRED	T&C acceptance is not required to offer VoWiFi service
		3 - IN PROGRESS	T&C capture and acceptance is on-going
TC_Operation (Conditional)	Integer	Returned only if TC_Status is 0 - NOT AVAILABLE	
		1 – WEBSHEET_IS_PREFERED	T&C capture and acceptance through web portal is the preferred option for the carrier. If device doesn't support this, it will take the other one (MSG), if available.
		2 – MSG_IS_PREFERED	T&C capture and acceptance through client is the preferred option for the carrier. If device doesn't support this, it will take the other one (WEBSHEET), if available.

Table 90. Entitlement Parameter - DCB T&C Status and Operation

11.1.3 DCB Service Parameters

During activation procedure of the DCB service, end-users could interact with Carrier Websheets or Device GUI to validate or approve some conditions. Both options are described in the following subsections.

These options (described in section 11.1.3.1 and 11.1.3.2) are not mutually exclusive. It means that both configurations could be provided to the device, and it will decide, based on its capabilities, which one to use.

11.1.3.1 DCB Client's Web Views Parameters

These are the parameters name and presence required in DCB.

- `ServiceFlow_URL`: Conditional

- `ServiceFlow_UserData`: Conditional
- `ServiceFlow_ContentsType`: Conditional

The entitlement parameters associated with the DCB service's web views are described in Table 91.

DCB Entitlement parameter	Type	Values	Description
ServiceFlow_URL (Conditional)	String	URL to a Service Provider site or portal	The URL of web views to be used by DCB client to present the user with DCB service management, which may include agreeing to the T&C of the DCB service.
ServiceFlow_UserData (Conditional)	String	Parameters or content to insert when invoking URL provided in the `ServiceFlow_URL` parameter	User data sent to the Service Provider when requesting the `ServiceFlow_URL` web view. It should contain user-specific attributes to improve user experience. The format must follow the `ServiceFlow_ContentsType` parameter. For content types of JSON and XML, it is possible to provide the base64 encoding of the value by preceding it with `encodedValue=`.
ServiceFlow_ContentsType (Conditional)	String	Specifies content and HTTP method to use when reaching out to the web server specified in `ServiceFlow_URL`.	
		NOT present	Method to `ServiceFlow_URL` is HTTP GET request with query parameters from `ServiceFlow_UserData`.
		json	Method to `ServiceFlow_URL` is HTTP POST request with JSON content from `ServiceFlow_UserData`.
		Xml	Method to `ServiceFlow_URL` is HTTP POST request with XML content from `ServiceFlow_UserData`.

Table 91. DCB Service Parameters - WebView Information

11.1.3.2 DCB Client's GUI Parameters

These are the parameters name and presence required in DCB.

- `MSG`: Conditional

The entitlement parameters associated with the DCB service's web views are described in Table *92*.

DCB Entitlement parameter	Type	Values	Description
MSG (Conditional)	Structure	multi-parameter value - see Table 93. DCB Service Parameters - GUI MSG Information for details	Specifies the message to be displayed/accepted/rejected through the client.

Table 92. DCB Service Parameters - Client Information

MSG object	Type	Description
Title (mandatory)	String	The window title where the user message is displayed.
Message (mandatory)	String	The message that is displayed to the user. Please note the message may contain references to HTTP addresses (websites) that need to be highlighted and converted into links by the device/client.
Accept_btn (mandatory)	String	This indicate whether an "Accept" button is shown with the message on device UI. The action associated with the Accept button on the device/client is to clear the message box. • "1" indicates that an "Accept" button shall be displayed. • "0" indicates that no "Accept" button shall be displayed.
Reject_btn (mandatory)	String	This indicate whether an "Decline" button is shown with the message on device UI. The action associated with the Reject button on the device/client is to revert the configured services to their defined default behaviour. • "1" indicates that a "Decline" button has to be displayed. • "0" indicates that no "Decline" button has to be displayed.

Table 93. DCB Service Parameters - GUI MSG Information

11.1.4 DCB Message for Incompatible Status

These are the parameters name and presence required in DCB for Incompatible status.

- `MessageForIncompatible`: Mandatory

When the status for the DCB entitlement is INCOMPATIBLE (see 11.1.1) and the end-user tries to activate DCB, the DCB client should show a message to the end-user indicating why activation was refused.

This entitlement parameter provides the content of that message, as decided by the Service Provider. Table 94 describes this DCB entitlement parameter.

DCB Entitlement parameter	Type	Description
MessageForIncompatible (Mandatory)	String	A message to be displayed to the end-user when activation fails due to an incompatible DCB Entitlement Status

Table 94. Entitlement Parameter - DCB Message for Incompatible Status

11.2 Client Behavior for DCB Entitlement Configuration

The entitlement parameters for DCB provide an overall status for the service as well as additional information associated with the activation procedure and provisioning of the service.

As such, the entitlement configuration for DCB carries information that impacts the behavior of the DCB client.

The client shall then activate (or deactivate) the DCB service according to the combination of the DCB's general setting on the device (controlled by the end-user) and the received DCB entitlement configuration.

The client shall also use the DCB entitlement parameters to decide if DCB web views for activation and service management should be presented to the end-user. This includes country-specific details on the need for DCB's Terms & Conditions acceptance and the requirement to enable or not the service - a country's regulations may require users to enable the service as well as agree to the Terms & Conditions of the service when activating DCB.

11.3 Entitlement Modes of DCB Client

To simplify the description of the client's behavior with respect to the DCB entitlement configuration, a set of "DCB entitlement modes" for the client is defined, each with specific expectations on the client side.

The relationship between the values of the DCB entitlement parameters and the DCB entitlement modes are shown in Table 95.

DCB Entitlement parameter		DCB Entitlement mode
Entitlement Status	TC Status	
INCOMPATIBLE	Any	Cannot purchase
DISABLED	NOT AVAILABLE	Service Data Missing
	AVAILABLE or NOT REQUIRED	Service Being Provisioned
ENABLED	AVAILABLE or NOT REQUIRED	Can purchase
PROVISIONING	Any	Service Being Provisioned

Table 95. DCB Entitlement Modes

The description of each DCB entitlement mode follows.

11.3.1 DCB Entitlement Mode – Cannot purchase.

The Client shall stay in this mode when:

- **`EntitlementStatus`** is INCOMPATIBLE

The Client cannot use the DCB service.

Due to end-user's action, the client may send a request to the Entitlement Configuration Server to refresh the DCB entitlement status. If the received status is still INCOMPATIBLE, the device shall either display **`MessageForIncompatible`** when it is not void, or the default device error message (if any).

11.3.2 DCB Entitlement Mode – Service Being Provisioned

There can be two scenarios where the client stays in this mode:

- **`EntitlementStatus`** is DISABLED
- **`TC_Status`** is AVAILABLE or NOT REQUIRED

The Client cannot use the DCB service.

Due to end-user's action, the Client may send a request to the Entitlement Configuration Server to refresh the DCB entitlement status. If the received status leads to the same mode, the Client shall open a web view and instruct the end-user to enter the required missing DCB service information (enable DCB).

- Or
  - **`EntitlementStatus`** is PROVISIONING

The Client cannot use the DCB service.

Due to end-user's action, the Client may send a request to the Entitlement Configuration Server to refresh the DCB entitlement status.

11.3.3 DCB Entitlement Mode – Service Data Missing

The Client shall stay in this mode when:

- **`EntitlementStatus`** is DISABLED
- **`TC_Status`** is NOT AVAILABLE

The Client cannot use the DCB service.

Due to end-user's action, the Client may send a request to the Entitlement Configuration Server to refresh the DCB entitlement status. If the received status leads to the same mode, the Client shall either open a web view or display a message and instruct the end-user to enter the required missing DCB service information (T&C).

11.3.4 DCB Entitlement Mode – Can purchase.

The Client shall stay in this mode when all the following conditions are met:

- **`EntitlementStatus`** is ENABLED
- **`TC_Status`** is AVAILABLE or NOT REQUIRED

When entering this mode, the client can use the DCB service.

11.4 DCB Flows

11.4.1 DCB Entitlement Request and Notifications

Figure 55 shows the standard entitlement request for DCB (steps 1 to 4). Additionally, it is added a refresh request triggered by an entitlement changed triggered by the carrier (steps 5 to 10).

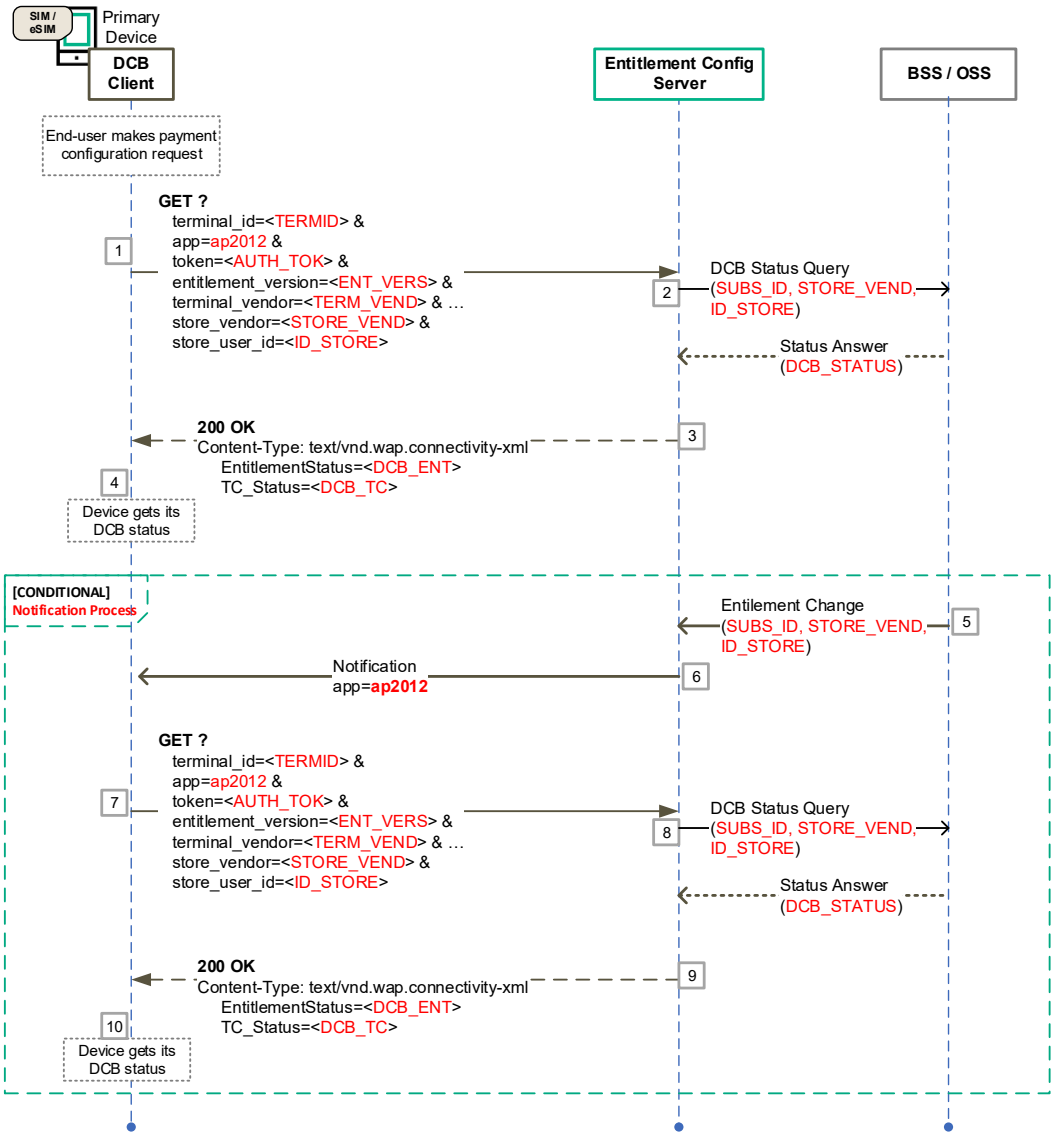

Figure 55. DCB Entitlement Request Flow and Notification Update

11.4.2 DCB Entitlement Request with user Interaction

Figure 56 explains how the user could interact with backend systems (through Websheets or Device GUI) to Accept or Reject, for example, some Terms&Conditions. Based on the ECS response (step 3), device will decide what's the preferred option: Websheets (steps 4 to 7) or through Device GUI (steps 8 to 11).

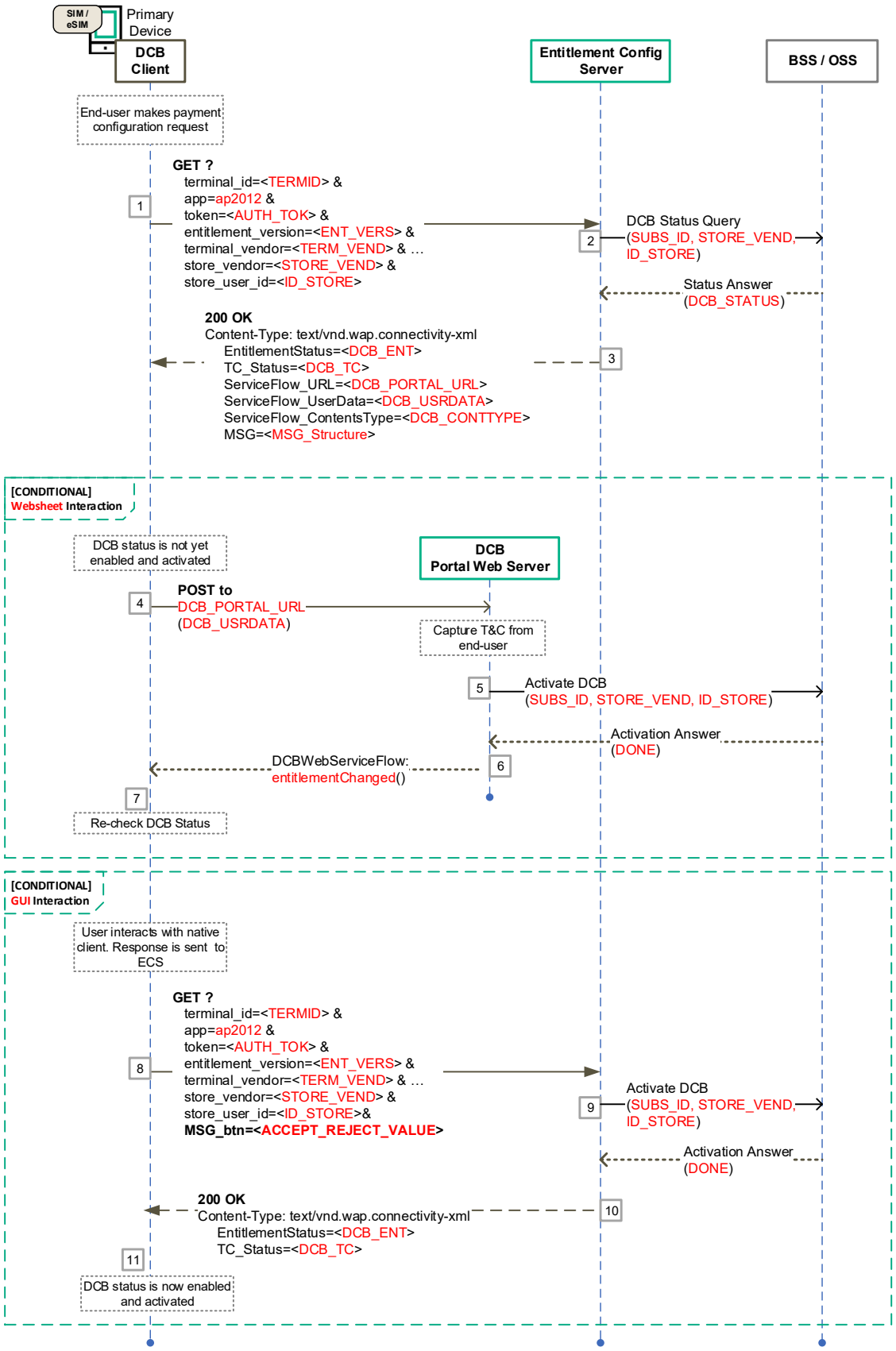

Figure 56. Entitlement Request Flow with User Interaction (Websheet or GUI)

11.5 DCB Request/Responses examples

11.5.1 Initial Requests

Initial request can use GET or POST methods.

Table 96 presents a sample HTTP GET request for DCB entitlement with the parameters located in the HTTP query string.

```
GET ? terminal_id = 013787006099944&
token = es7w1erXjh%2FEC%2FP8BV44SBmVipg&
terminal_vendor = TVENDOR&
terminal_model = TMODEL&
terminal_sw_version = TSWVERS&
entitlement_version = ENTVERS&
app = ap2012&
store_vendor=STORE_VEND&
store_user_id=<STORE_USR>&
vers = 1 HTTP/1.1

Host: entitlement.telco.net:9014
User-Agent: PRD-TS43 TVENDOR/TMODEL IMS-Entitlement/TSWVERS OS-Android/8.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Table 96. Example of an HTTP GET Entitlement Configuration Request for DCB

Table 97 presents a sample HTTP POST request for DCB entitlement with the parameters located in the HTTP message body.

```
POST / HTTP/1.1
Host: entitlement.telco.net:9014
User-Agent: PRD-TS43 TVENDOR/TMODEL IMS-Entitlement/TSWVERS OS-Android/8.0Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Content-Type: application/json

{
      "terminal_id" : "013787006099944",
      "entitlement_version" : "ENTVERS",
      "token" : "es7w1erXjh%2FEC%2FP8BV44SBmVipg",
       "terminal_vendor" : "TVENDOR",
       "terminal_model" : "TMODEL",
       "terminal_sw_version" : "TSWVERS",
       "app" : "ap2012",
       "store_vendor" : "STORE_VEND",
       "store_user_id" : "STORE_USR",
       "vers" : "1"
}
```

Table 97. Example of an HTTP POST Entitlement Configuration Request for DCB

11.5.2 Initial Responses

Table 98 presents an example for a returned DCB entitlement configuration in XML format where entitlement is enabled, and T&C is not required.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS"
        <parm name="version" value="1"/>
        <parm name="validity" value="172800"/>
    </characteristic>
     <characteristic type="TOKEN">
        <parm name="token" value="ASH127AHHA88SF"/>
    </characteristic>
     <characteristic type="APPLICATION">
        <parm name="AppID" value="ap2012"/>
        <parm name="EntitlementStatus" value="1"/>
        <parm name="TC_Status" value="2"/>
    </characteristic>
</wap-provisioningdoc>
```

Table 98. DCB configuration response in XML format example where DCB is enabled, and T&C is not required.

Table 99 presents an example for a returned Data Plan Information entitlement configuration in JSON format where entitlement is enabled, and T&C is not required.

```
{
    "Vers" : {
        "version" : "1",
        "validity" : "172800"
    },
    "Token" : {              // Optional
        "token" : "ASH127AHHA88SF"
    },
    "ap2012": {            // DCB Entitlement settings
        "EntitlementStatus" : 1,
        "TC_Status" : 2
    }
}
```

Table 99. DCB configuration response in JSON format example where DCB is enabled, and T&C is not required.

Table 100 presents an example for a returned DCB entitlement configuration in XML format where entitlement is enabled, and T&C is available.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS"
        <parm name="version" value="1"/>
        <parm name="validity" value="172800"/>
    </characteristic>
     <characteristic type="TOKEN">
        <parm name="token" value="ASH127AHHA88SF"/>
    </characteristic>
     <characteristic type="APPLICATION">
        <parm name="AppID" value="ap2012"/>
        <parm name="EntitlementStatus" value="1"/>
        <parm name="TC_Status" value="1"/>
    </characteristic>
</wap-provisioningdoc>
```

Table 100. DCB configuration response in XML format example where DCB is enabled, and T&C is available.

Table 101 presents an example for a returned DCB entitlement configuration in JSON format where entitlement is enabled, and T&C is available.

```
{
    "Vers" : {
        "version" : "1",
        "validity" : "172800"
    },
    "Token" : {               // Optional
        "token" : "ASH127AHHA88SF"
    },
    "ap2012": {            // DCB Entitlement settings
        "EntitlementStatus" : 1,
        "TC_Status" : 1
    }
}
```

Table 101. DCB configuration response in JSON format example where DCB is enabled, and T&C is available.

Table 102 presents an example for a returned DCB entitlement configuration in XML format where DCB entitled and T&C Status “NOT AVAILABLE” and ECS provides both, Websheets and GUI, options to interact with the user. `TC_Operation=1` identifies that Websheet is the preferred one for the carrier.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS"
        <parm name="version" value="1"/>
        <parm name="validity" value="172800"/>
    </characteristic>
     <characteristic type="TOKEN">
        <parm name="token" value="ASH127AHHA88SF"/>
    </characteristic>
     <characteristic type="APPLICATION">
        <parm name="AppID" value="ap2012"/>
        <parm name="EntitlementStatus" value="1"/>
        <parm name="TC_Status" value="0"/>
        <parm name="TC_Operation" value="1"/>
        <parm name="ServiceFlow_URL" value=" https://www.MNO.org/termsAndCons"/>
        <parm name="ServiceFlow_UserData" value="encodedValue=eyJpbXNpIjo…OiJ"/>
        <parm name="ServiceFlow_ContentsType" value="json"/>
        <characteristic type="MSG">
            <parm name="title" value="Terms and Conditions"/>
            <parm name="message" value="Are you agree with …"/>
            <parm name="Accept_btn" value="1"/>
            <parm name="Reject_btn" value="0"/>
        </characteristic>
    </characteristic>
</wap-provisioningdoc>
```

Table 102. DCB configuration response in XML format example providing Websheet and GUI parameters.

Table 103 presents an example for a returned DCB entitlement configuration in XML format where DCB entitled and T&C Status “NOT AVAILABLE” and ECS provides both, Websheets and GUI, options to interact with the user. `TC_Operation=2` identifies that GUI is the preferred one for the carrier.

```
{
    "Vers" : {
        "version" : "1",
        "validity" : "172800"
    },
    "Token" : {              // Optional
        "token" : "ASH127AHHA88SF"
    },
    "ap2012": {            // DCB Entitlement settings
        "EntitlementStatus": 1,
        "TC_Status" : 0,
        "TC_Operation" : 1,
        "ServiceFlow_URL": "https://www.MNO.org/termsAndCons",
        "ServiceFlow_UserData": "encodedValue=eyJpbXNpIjo…OiJ",
        "ServiceFlow_ContentsType": "json"
        "MSG": {
            "title": "Terms and Conditions",
            "message": "Are you agree with …",
            "Accept_btn": 1,
            "Reject_btn": 0
        }

    }
}
```

Table 103. DCB configuration response in JSON format example providing Websheet and GUI parameters.

Table 104 presents an example for a returned DCB entitlement configuration in XML format where entitlement is incompatible.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS"
        <parm name="version" value="1"/>
        <parm name="validity" value="172800"/>
    </characteristic>
     <characteristic type="TOKEN">
        <parm name="token" value="ASH127AHHA88SF"/>
    </characteristic>
     <characteristic type="APPLICATION">
        <parm name="AppID" value="ap2012"/>
        <parm name="EntitlementStatus" value="2"/>
        <parm name="MessageForIncompatible" value="Sorry your MNO have no Carrier
Billing"/>
    </characteristic>
</wap-provisioningdoc>
```

Table 104. DCB configuration response in XML format example where DCB is incompatible.

Table 105 presents an example for a returned DCB entitlement configuration in JSON format where entitlement is incompatible.

```
{
    "Vers" : {
        "version" : "1",
        "validity" : "172800"
    },
    "Token" : {               // Optional
        "token" : "ASH127AHHA88SF"
    },
    "ap2012": {            // DCB Entitlement settings
        "EntitlementStatus" : 2,
        "MessageForIncompatible" : "Sorry your MNO have no Carrier Billing"
    }
}
```

Table 105. DCB configuration response in JSON format example where DCB is incompatible.

Table 106 presents an example for a returned DCB entitlement configuration in XML format where DCB not entitled, and service flow required.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS"
        <parm name="version" value="1"/>
        <parm name="validity" value="172800"/>
    </characteristic>
     <characteristic type="TOKEN">
        <parm name="token" value="ASH127AHHA88SF"/>
    </characteristic>
     <characteristic type="APPLICATION">
        <parm name="AppID" value="ap2012"/>
        <parm name="EntitlementStatus" value="0"/>
        <parm name="TC_Status" value="0"/>
        <parm name="TC_Operation" value="1"/>
        <parm name="ServiceFlow_URL" value="https://www.MNO.org/entDisabled"/>
        <parm name="ServiceFlow_UserData" value="encodedValue=eyJpbXNpIjo…OiJ"/>
        <parm name="ServiceFlow_ContentsType" value="json"/>
    </characteristic>
</wap-provisioningdoc>
```

Table 106. DCB configuration response in XML format example where DCB not entitled, and service flow required.

Table 107 presents an example for a returned DCB entitlement configuration in JSON format where DCB not entitled, and service flow required.

```
{
    "Vers" : {
        "version" : "1",
        "validity" : "172800"
    },
    "Token" : {              // Optional
        "token" : "ASH127AHHA88SF"
    },
    "ap2012": {            // DCB Entitlement settings
        "EntitlementStatus": 0,
        "TC_Status": 0,
        "TC_Operation": 1;
        "ServiceFlow_URL": "https://www.MNO.org/entDisabled",
        "ServiceFlow_UserData": "encodedValue=eyJpbXNpIjo…OiJ",
        "ServiceFlow_ContentsType": "json"
    }
}
```

Table 107. DCB configuration response in JSON format example where DCB not entitled, and service flow required.

11.6 DCB Client Considerations around Web View Callbacks

11.6.1 entitlementChanged() Callback function

The **`entitlementChanged()`** callback function indicates that the DCB service flow ended properly between the device and DCB portal web server.

The web view to the end-user should be closed and the DCB client shall make a request for the latest DCB entitlement configuration status, via the proper TS.43 entitlement configuration request.

Based on the returned set of status parameters, the DCB client shall behave as specified in section 11.2

In Figure 55 shows, in step 6 how the **`entitlementChanged()`** callback function fits into the typical steps involved with DCB entitlement configuration.

11.6.2 dismissFlow() Callback function

The **`dismissFlow()`** callback function indicates that the DCB service flow ends prematurely, either caused by user action (DISMISS button for example) or by an error in the web sheet logic or from the network side.

As a result of the dismissal of the service flow, the DCB entitlement status has not been updated by the DCB portal.

The web view to the end-user should be closed and the DCB client should not make a request for the latest DCB entitlement configuration status.

12 Private User Identity

Private User Identity (from here on out Private UserID) use case allows devices to connect to Access Points using SIM-based authentication. EAP methods are used for this purpose.

As per the SIM-based EAP Authentication, the device needs to connect to the carrier network to perform to validate the credentials. For doing this, on the first ever connection to such a Wi-Fi network (for the EAP-Request/Identity & EAP-Response/Identity messages), the peer must provide its permanent subscriber identity information (IMSI) to the authenticator. This identity is sent in the clear.

This use case will not only solve the identity encryption for the first connection to the Wi-Fi network but also validate if a specific user is eligible or not to use this type of service.

Figure 57 presents the high-level architecture of the Private UserID use case.

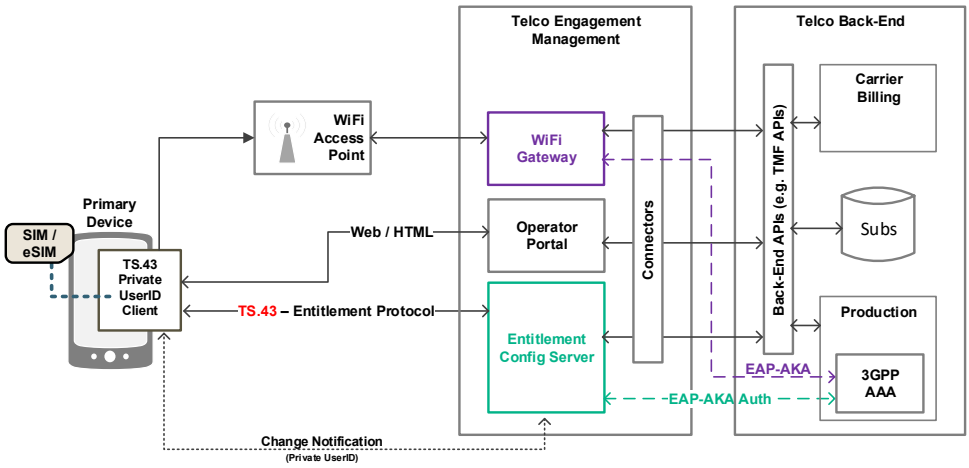

Figure 57. Private User ID high-level architecture

12.1 Private UserID entitlement parameters

Parameters for the Private UserID entitlement provide the overall status of the Private UserID service to the client, as well as the different sub-status associated with the activation procedure of the service.

The Private UserID entitlement parameters also include information associated with the web views presented to users by the Private UserID client during activation and management of the service.

12.1.1 Private UserID Entitlement Status

This is the parameter name and presence required in Private UserID.

- `EntitlementStatus`: Mandatory

This parameter indicates the overall status of the Private UserID entitlement, stating if the service can be offered on the device, and if it can be activated or not by the end-user.

The different values for the Private UserID entitlement status are provided in Table 108

Private UserID Entitlement parameter	Type	Values	Description
EntitlementStatus (Mandatory)	Integer	0 - DISABLED	Private UserID service not entitled
		1 - ENABLED	Private UserID service entitled
		2 - INCOMPATIBLE	Private UserID service cannot be offered

Table 108. Entitlement Parameter - Private UserID Overall Status

12.1.2 Private UserID Data

These are the parameters name and presence required in Private UserID for Encoded Data

- `PrivateUserID`: Conditional
- `PrivateUserIDType`: Mandatory if `PrivateUserID` is present.
- `PrivateUserIDExpiry`: Optional

The following parameters describe the information to be shared with the device. Initially, Private UserID use case only strictly requires **`IMSI`** (for EAP-AKA authentication) to be encoded, but there could be any other info as part of the `PrivateUserID` parameter if required by the WiFi Gateway.

Private UserID Entitlement parameter	Type	Values	Description
PrivateUserID	String	Any valid string. It could be an empty string for the `PrivateUserIDType` =1	Present if **`EntitlementStatus`** is "1". Encoded information to be sent to the device for devices usage. See section 12.4 for special considerations. It is possible to provide the base64 encoding of the value by preceding it with `encodedValue=`
PrivateUserIDType	Integer	Defines the type of data includes in the `PrivateUserID` parameter.	
		1 – PSEUDONYM	Used when the `AT_NEXT_PSEUDONYM` in the EAP-Request/AKA-Challenge is defined as `PrivateUserID`.
		2 – OTHER	Used when the content in the `PrivateUserID` parameter includes an encrypted data (including IMSI). For additional info see section 12.4

Private UserID Entitlement parameter	Type	Values	Description
PrivateUserIDExpiry (Optional)	Time	in ISO 8601 format, of the form YYYY-MM-DDThh:mm:ssTZD	The time/date when the `PrivateUserID` expires and should be renewed by the device.

Table 109. Entitlement Parameter – Private UserID Data

NOTE.- There are some interactions in the end-to-end Private UserID Authentication flow, that are out of scope of this document (TS.43). Section 12.4 provides some considerations about how the info could be managed.

12.2 Private UserID Flows

Private UserID Flows don't differ a lot from the VoWiFi or Direct Carrier Billing use cases.

Figure 58 shows an initial request (requiring a Full Authentication) where ECS interacts with the AAA. This flow is the standard one for a Full Authentication process as described in Figure 2 (2.8.1), but at the end of the flow, ECS will send the proper parameters for the Private UserID use case.

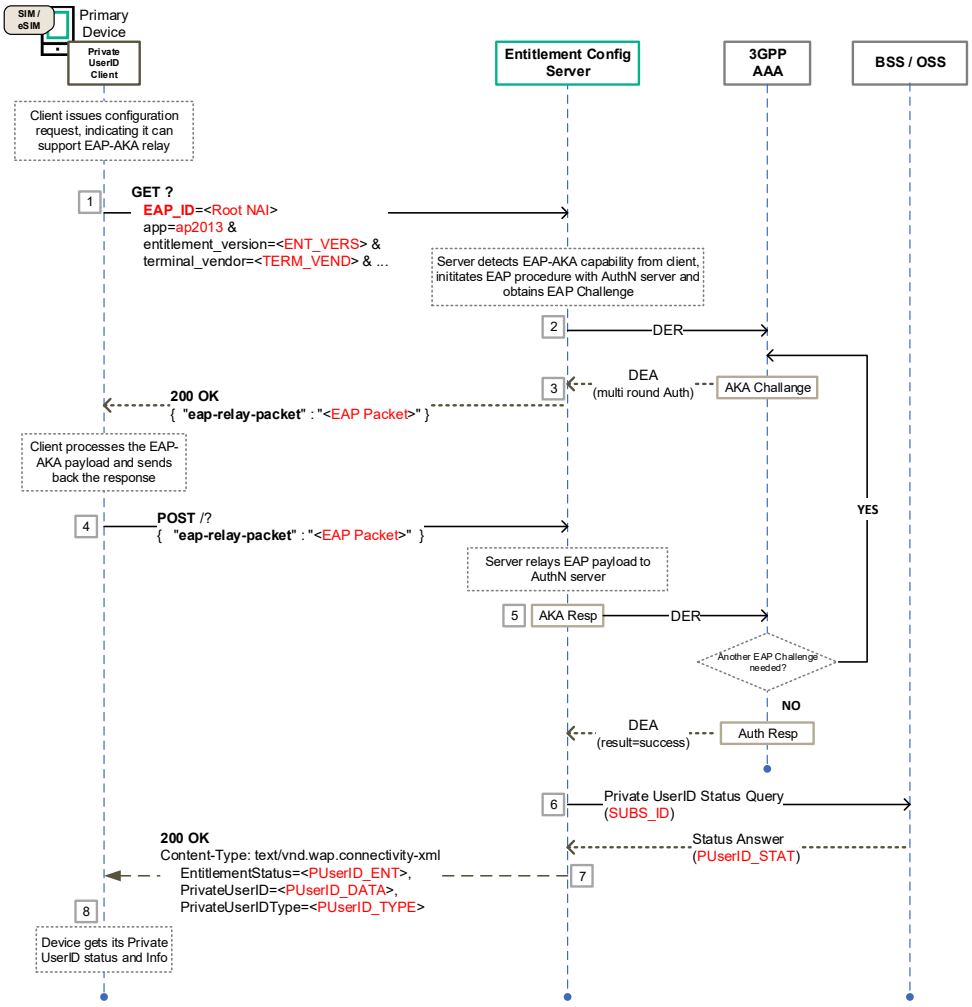

Figure 58. Private User ID Entitlement Request with Full Authentication

Figure 59 shows the standard entitlement request for Private UserID (steps 1 to 4) when UE already has an authentication token. Additionally, it is added a refresh request triggered by an entitlement changed triggered by the carrier (steps 5 to 10).

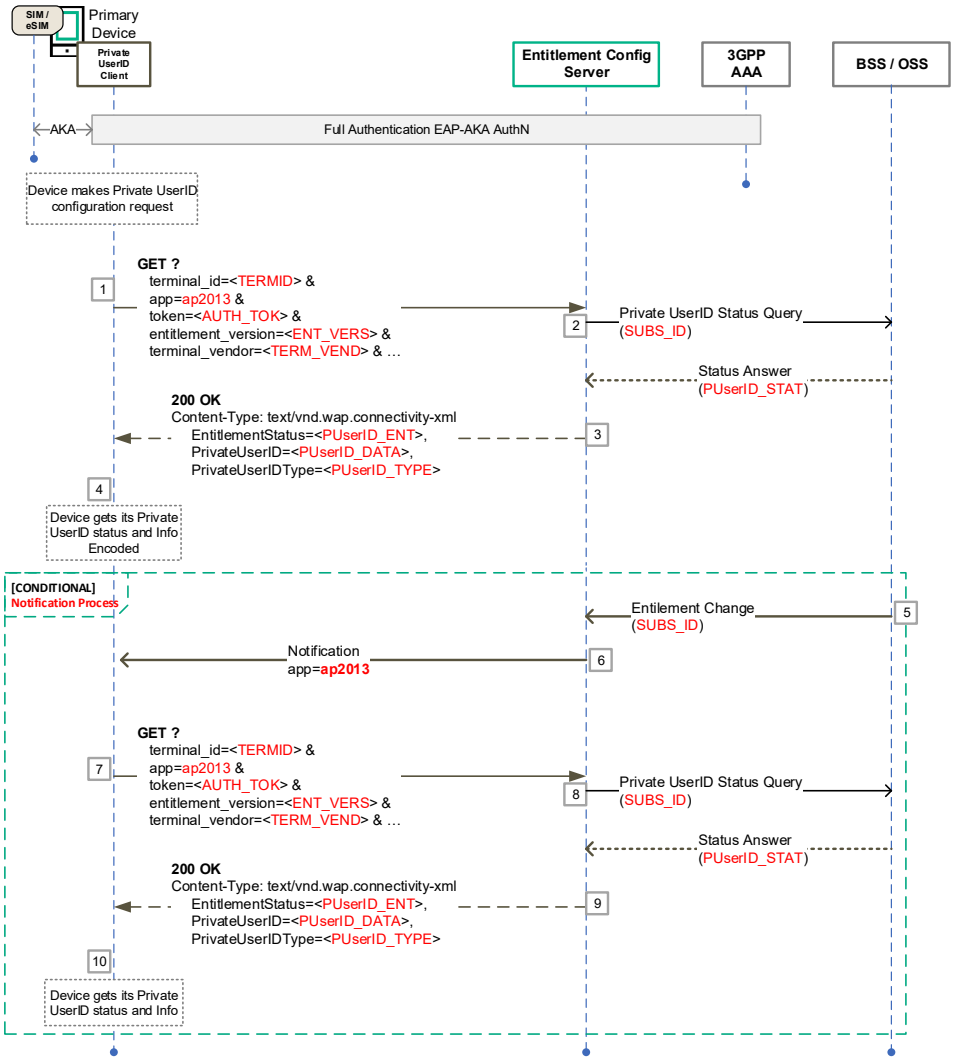

Figure 59. Private User ID Entitlement Request Flow and Notification Update

12.3 Private UserID Request/Responses examples

12.3.1 Initial Requests

Initial request can use GET or POST methods.

Table 110 presents a sample HTTP GET request for Private UserID entitlement with the parameters located in the HTTP query string.

```
GET ? terminal_id = 013787006099944&
token = es7w1erXjh%2FEC%2FP8BV44SBmVipg&
terminal_vendor = TVENDOR&
terminal_model = TMODEL&
terminal_sw_version = TSWVERS&
entitlement_version = ENTVERS&
app = ap2013&
vers = 1 HTTP/1.1

Host: entitlement.telco.net:9014
User-Agent: PRD-TS43 TVENDOR/TMODEL IMS-Entitlement/TSWVERS OS-Android/8.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Table 110. Example of an HTTP GET Entitlement Configuration Request for Private UserID

Table 111 presents a sample HTTP POST request for Private UserID entitlement with the parameters located in the HTTP message body.

```
POST / HTTP/1.1
Host: entitlement.telco.net:9014
User-Agent: PRD-TS43 TVENDOR/TMODEL IMS-Entitlement/TSWVERS OS-Android/8.0Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Content-Type: application/json

{
      "terminal_id" : "013787006099944",
      "entitlement_version" : "ENTVERS",
      "token" : "es7w1erXjh%2FEC%2FP8BV44SBmVipg",
       "terminal_vendor" : "TVENDOR",
       "terminal_model" : "TMODEL",
       "terminal_sw_version" : "TSWVERS",
       "app" : "ap2013",
       "vers" : "1"
}
```

Table 111. Example of an HTTP POST Entitlement Configuration Request for Private UserID

12.3.2 Initial Responses

Table 112 presents an example for a returned Private UserID entitlement configuration in XML format where entitlement is enabled.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS"
        <parm name="version" value="1"/>
        <parm name="validity" value="172800"/>
    </characteristic>
     <characteristic type="TOKEN">
        <parm name="token" value="ASH127AHHA88SF"/>
    </characteristic>
     <characteristic type="APPLICATION">
        <parm name="AppID" value="ap2013"/>
        <parm name="EntitlementStatus" value="1"/>
        <parm name="PrivateUserID" value="RRHAXHQXFZivBlEOr2ZnlTbnn78xdrW5i"/>
        <parm name="PrivateUserIDType" value="2"/>
    </characteristic>
</wap-provisioningdoc>
```

Table 112. Private UserID configuration response in XML format example where Private UserID is entitled.

Table 113 presents an example for a returned Private UserID entitlement configuration in JSON format where entitlement is enabled.

```
{
    "Vers" : {
        "version" : "1",
        "validity" : "172800"
    },
    "Token" : {              // Optional
        "token" : "ASH127AHHA88SF"
    },
    "ap2013": {            // Private UserID Entitlement settings
        "EntitlementStatus" : 1,
        "PrivateUserID" : "RRHAXHQXFZivBlEOr2ZnlTbnn78xdrW5i",
        "PrivateUserIDType" : "2"
    }
}
```

Table 113. Private UserID configuration response in JSON format example where Private UserID is entitled.

12.4 Private UserID - Special considerations

TS.43 document only defines how devices and ECS interacts each other, as part of and specific use case. It's out of scope of this document to describe in detail how the ECS interacts with the carrier backend or how the device manages the info received by the ECS to use a specific service.

For the Private UserID, it is necessary to bear in mind the following considerations.

- In those cases where ECS is not able to provide `AT_NEXT_PSEUDONYM`, but the pseudonym usage (`PrivateUserIDType`=1) is the desired option, ECS will send an empty string as the `PrivateUserID` value, and the UE will be responsible to extract from `AT_ENCR_DATA` in the EAP-Request/AKA-Challenge.

- IMSI encrypted value will be sent in the `AT_IDENTITY` parameter, as part of the EAP-AKA/Identity-Response. The size of this Identity must smaller than 1016 bytes as defined in [18].

- ECS and WiFi Gateway must have an agreement in advance to know how to encrypt or decrypt the info. For simplicity it is recommended to use **JSON Web Tokens** (JWT) with **offline validation** to avoid overload in other systems. JWT are flexible enough to add more parameters without any big change.
  Encryption/Decryption and how the information is 'encapsulated' is out of scope of TS.43, but it is part of the E2E flow.

- To avoid any type of interoperability issues, and to make sure that the UE and Authentication Server derive the same MK, the following should be implemented.

  - **`PrivateUserIDType=1`**: Pseudonym will be used as Identity for deriving the MK = SHA1(Identity | IK | CK)

  - **`PrivateUserIDType=2`**: IMSI will be used as Identity for deriving the MK = SHA1(Identity | IK | CK)

- WiFi Gateway could implement their own Fast Re-Authentication process. This is out of scope of this document, and it is totally separate to the ECS Fast Authentication process defined in section 2.8.5.

13 Device and User Information

13.1 Phone Number Information

Phone Number (MSISDN) is one of the main subscription identifiers and it is required for multiple services. Although RCC.14 (reference [5]) implements the MSISDN parameter as part of the USER characteristic, this TS.43 specification provides additional alternatives to provide this information. These options are described in the following sections where the phone number could be requested by the primary device or by a different one (including an application server) which use a temporary token for validating the request.

In the case where the MSISDN should be encrypted, it is recommended to use base64 encoding, using `encodedValue` tag as described in Table 49. The procedure used to encrypt the MSISDN by ECS and decrypt by primary device or application server is out of scope of TS.43.

13.1.1 Phone Number Information from device

The device, as it does with other services like VoWiFi (section 3), VoCellular (section 4) or SMSoIP (section 5), will trigger a request to get MSISDN as part of its configuration. The main difference is that Phone Number is not considered as a specific service, so it doesn't require any specific entitlement validation.

Figure 60 presents the flow describing how the primary device triggers a `GetPhoneNumber` request. The steps are:

1. Authentication of the end-user by the SP's 3GPP AAA server is performed using proper EAP-AKA exchanges (see 2.8.1 for details).
2. The primary TS.43 client application makes a `GetPhoneNumber` request to the ECS.
3. ECS then queries the SP's back-end system managing the subscriptions to request the MSISDN assigned to the end-user.

   Otherwise, the ECS should reply with a 405 (see Table 13 for details) and end the flow there.
4. The SP's backend-end system answers to the ECS query, including the MSISDN in the case of a positive response.

The ECS generates a 200 OK response including the MSISDN and sends it to the device.

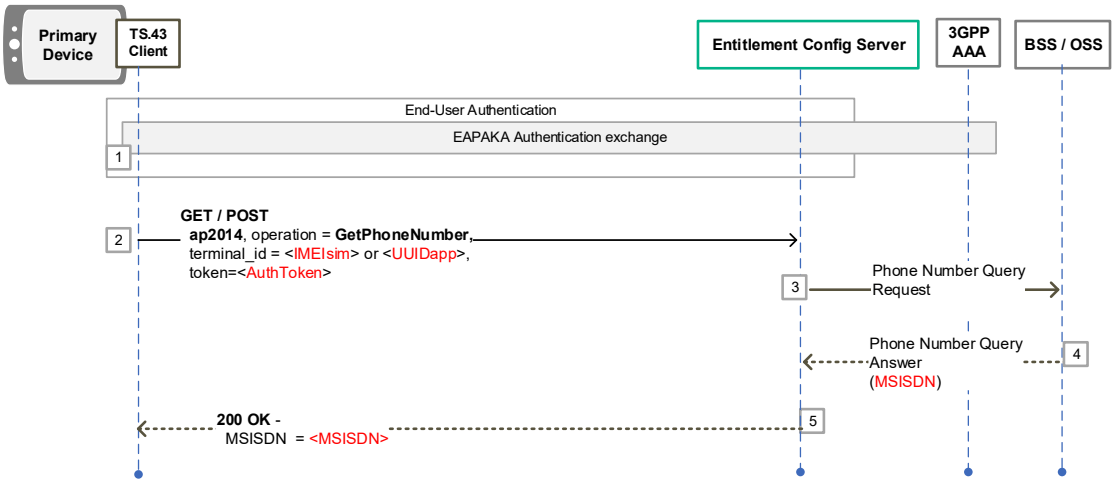

Figure 60. Phone Number Information from Primary Device

13.1.2 Phone Number Information through Application Server

The following presents the case where:

- The Primary ODSA client application is allowed for the type of primary device and enabled for the end-user (entitled).
- The Application Server can receive TemporaryToken from the ODSA client.
- The Application Server is authorized by the Entitlement Configuration Server to execute GET PHONE NUMBER operations on behalf of the ODSA client, using a TemporaryToken.

Figure 61 presents a call flow where the Application Server requests a Phone Number from the ECS on behalf of the ODSA client, using a TemporaryToken. Authentication (e.g. EAP-AKA, SMS-OTP) is performed before starting this procedure described in Figure 61.

1. Authentication of the end-user by the SP's 3GPP AAA server is performed using proper EAP-AKA exchanges (see 2.8.1 for details).
2. The Primary ODSA client application makes an **AcquireTemporaryToken** request to the ECS for a GetPhoneNumber operation target.
3. The ECS generates a 200 OK response with a `TemporaryToken`, a `TemporaryTokenExpiry` and the allowed GetPhoneNumber target operation.
4. The Primary ODSA client application sends the `TemporaryToken` alongside with the `TemporaryTokenExpiry` to the Application Server.
   Note that the communication between the device (client) and the application server is outside the scope of TS.43 and it only appears in the flow as 'Informative'.

If Server to Server authentication mechanism is implemented, it should be necessary to follow step 5, otherwise move directly to step 6:

5. Optional - The Application Server requests an Access Token to the Authorization Server controlling the access to the ECS.
6. The Application Server makes a **GetPhoneNumber** request to the ECS, including the `TemporaryToken` in the parameters, while also ensuring the

`TemporaryTokenExpiry` has not been exceeded. Optional - In the case where a Server-to-Server authentication is required, the latest and still valid `access_token` and `requestor_id` are to be included in the request.

7. The ECS authenticates the **GetPhoneNumber** request based on the `TemporaryToken` and evaluates if the requested operation is authorized. This evaluation could be based on various information such as `requestor_id` or device info in request parameters.
   If the operation is authorized, the ECS then queries the SP's back-end system managing the subscriptions to request the MSISDN assigned to the end-user. Otherwise, the ECS should reply with a 403 (see 2.8.6 for details) and end the flow there.
8. The SP's backend-end system answers to the ECS query, including the `MSISDN` in the case of a positive response.

The ECS generates a 200 OK response including the `MSISDN` and sends it to the Application Server.

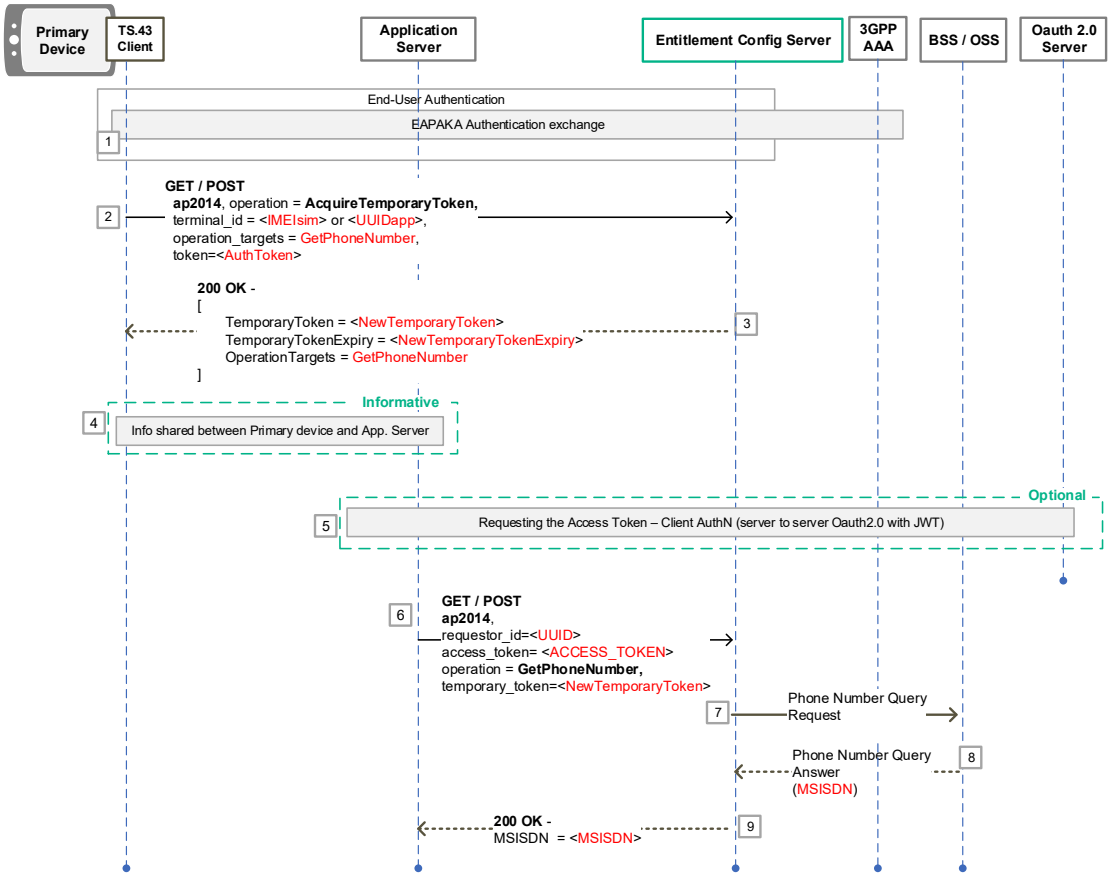

Figure 61. Phone Number Information through Application Server

13.2 Phone Number Verification

There could be some scenarios where device, 3rd party application or Application server, would need to verify that the phone number (MSISDN) it has, it's the right one. For doing this

operation, instead of pulling (getting) the information (MSISDN) from the ECS, it will be pushed for validation.

ECS will validate that the MSISDN included in the `msisdn` parameter for the `VerifyPhoneNumber` operation, is the same phone number that the one associated to the authentication method belonging to the device.

This Authentication method could be any of the ones described in this document (EAP-AKA, OIDC, SMS OTP …, see section 2.8 for detailed info) or any other token derived from that Authentication (for example TemporaryToken).

Figure 62 presents a call flow where the device (could be also a 3rd Party App installed in the device) requests a Phone Number Verification with a token acquired after some of the AuthN methods.

1. Device (or 3rd Party App) performs the Authentication (any of the supported by TS.43) and receives an Authentication Token from ECS.
2. Device (or 3rd Party App) sends `VerifyPhoneNumber` request including the MSISDN for validation.
3. ECS, using the token, gets the device identifier and interacts with the backend to get the MSISDN associated to that identifier.
4. Backend provides the MSISDN requested by ECS.
5. ECS compares the MSISDN receive by the backend with the one in the request.
6. ECS sends the result of the comparation to the device. Optionally, and if the validation is success, ECS can send the MSISDN that have been validated.

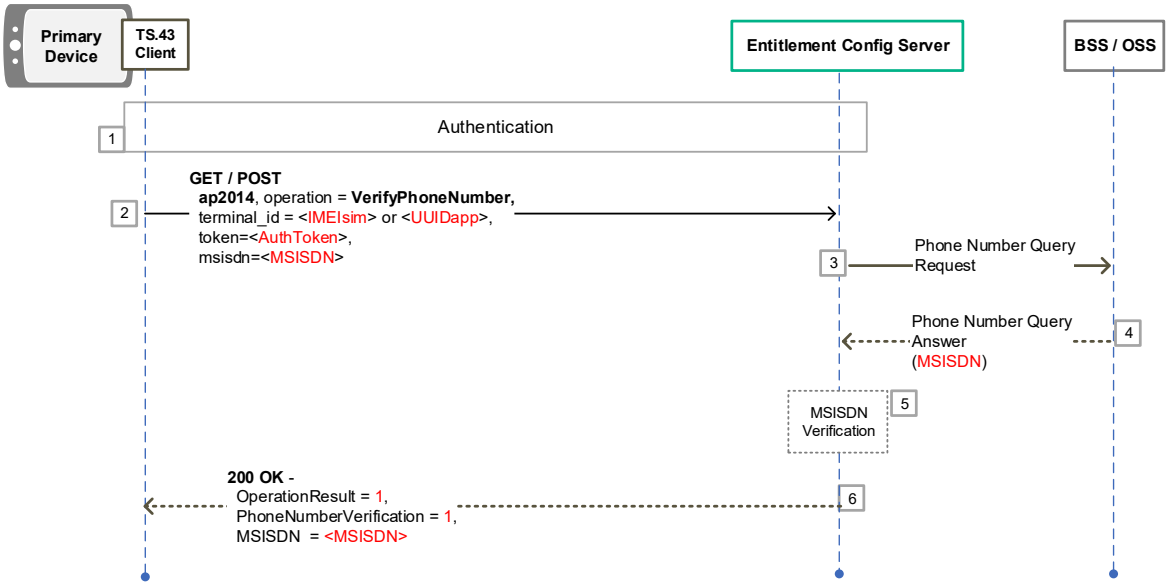

Figure 62.PhoneNumber Verification Flow

13.3 Subscriber Information

Subscriber information includes Phone Number (MSISDN) which is one of the main subscription identifiers as well as other information such as identifiers for the SIM, identifiers for the MVNO etc. This TS.43 specification provides an alternative to provide this information

that could be requested by an application server which uses a temporary token for validating the request.

13.3.1 Subscriber Information through Application Server

The following presents the case where:

- The Primary ODSA client application is allowed for the type of primary device and enabled for the end-user (entitled).
- The Application Server can receive TemporaryToken from the ODSA client.
- The Application Server is authorised by the Entitlement Configuration Server to execute `GetSubscriberInfo` operations on behalf of the ODSA client, using a TemporaryToken.

Figure 63 presents a call flow where the Application Server requests Subscriber Information from the ECS on behalf of the ODSA client, using a TemporaryToken. Authentication (e.g. EAP-AKA) is performed before starting this procedure, as described in Figure 61.

1. Authentication of the end-user by the SP's 3GPP AAA server is performed using proper EAP-AKA exchanges (see 2.8.1 for details).
2. The Primary ODSA client application makes an **AcquireTemporaryToken** request to the ECS for a **GetSubscriberInfo** operation target.
3. The ECS generates a 200 OK response with a TemporaryToken, a TemporaryTokenExpiry and the allowed target operation.
4. The Primary ODSA client application sends the TemporaryToken alongside with the TemporaryTokenExpiry to the Application Server.

   Note that the communication between the device (client) and the Application Server is outside the scope of TS.43 and it only appears in the flow as ‘Informative’.

If Server to Server authentication mechanism is implemented, it should be necessary to follow step 5, otherwise move directly to step 6:

5. Optional - The Application Server requests an Access Token to the Authorization Server controlling the access to the ECS.
6. The Application Server makes a **GetSubscriberInfo** request to the ECS, including the TemporaryToken in the parameters, while also ensuring the TemporaryTokenExpiry has not been exceeded. Optional - In the case where a Server-to-Server authentication is required, the latest and still valid access_token and requestor_id are to be included in the request.
7. The ECS authenticates the **GetSubscriberInfo** request based on the TemporaryToken and evaluates if the requested operation is authorized. This evaluation could be based on various information such as requestor_id or device info in request parameters.

If the operation is authorized, the ECS then queries the SP's back-end system managing the subscriptions to request the Subscriber Information for the end-user.

Otherwise, the ECS should reply with a 403 (see 2.8.6 for details) and end the flow there.

8. The SP's backend-end system answers to the ECS query in the case of a positive response.

The ECS generates a 200 OK response including the parameters detailed in section 6.5.X and sends it to the Application Server.

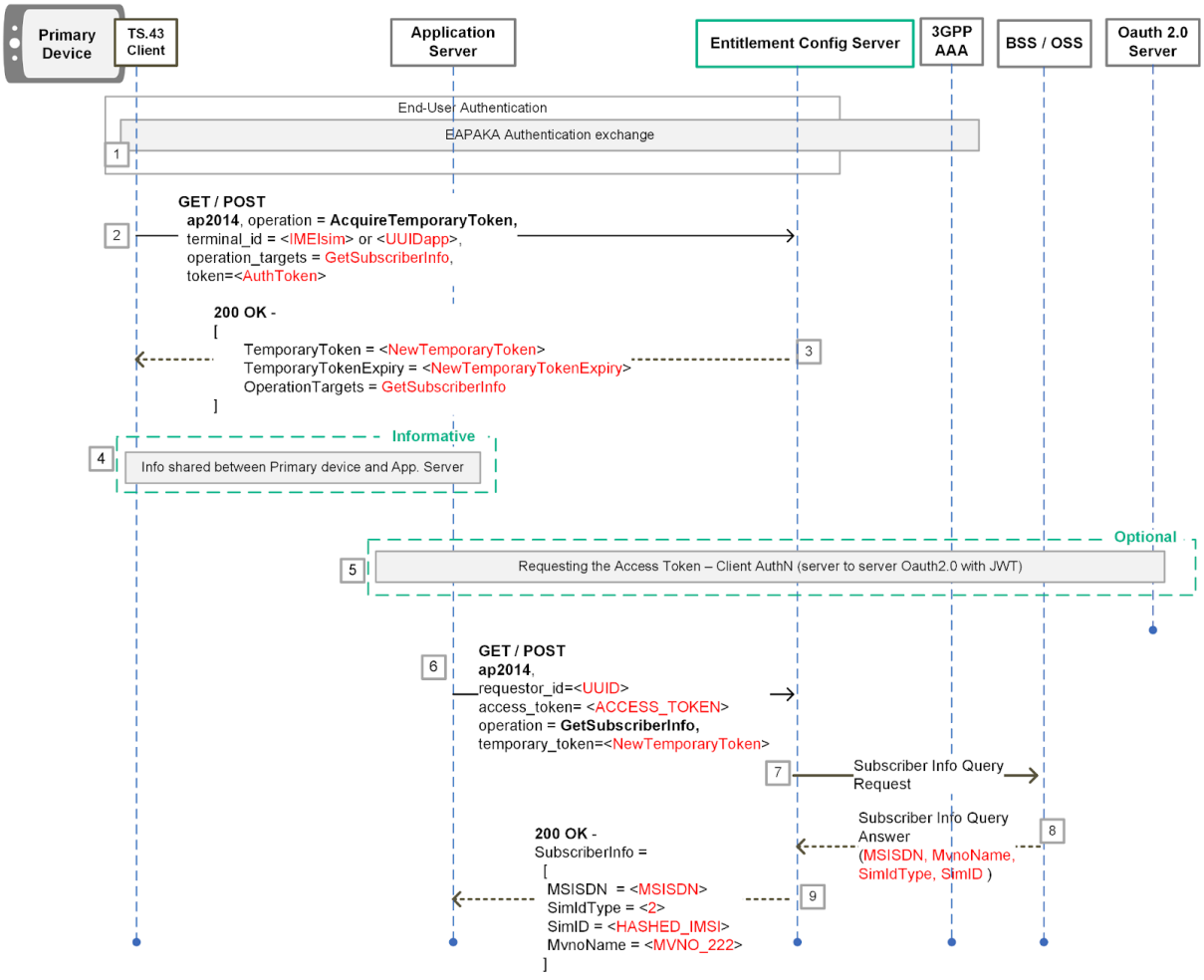

Figure 63. Subscriber Information through Application Server

14 Device App authentication

This section describes different use cases where the operator provides a token to authenticate subscribers and/or applications installed in the subscriber´s device.

- Operator token use case
- App token use case

For both cases:

- A 3rd party application is installed on the device and is not capable of performing TS.43 functions.
- The TS.43 client is able to call TS.43 methods, including EAP_AKA authentication.
- The 3rd party App and the ECS previously exchanged a shared security information to enable authentication. This could be done directly between ECS and app-backend or with a dedicated partner management system e.g. as described in GSMA IDG specifications.
- The 3rd party App is authorized to request app authentication from the TS.43 client via device-specific access control.
- Note: ECS may implement additional features to enable enhanced access control mechanisms (e.g. Auth Server)

For these use cases the appID ap2015 is used in the requests.

14.1 Operator Token use case

The benefit of this procedure is that an app can gain SIM based authentication, without requiring access to the EAP-AKA token or the temporary_token, so that the security and integrity of the tokens is maintained.

Note.- When access_token (eligibility token in ASAC.01) is present in the request, it may be more efficient to validate this token in the first stage before executing the standard authentication actions (Full or Fast Authentication).

14.1.1 Device App authentication Request Parameters

For device App authentication, devices require additional parameters in the HTTP requests, outside of the ones described in 2.3 and 6.2. Table 114 presents the new parameters and their associated operations.

New GET parameters app authentication	Type	Values	Description
operation	String	`AcquireOperatorToken`	Indicates the operation requested by the TS.43 client

New GET parameters app authentication	Type	Values	Description
access_token (optional)	String	Used by the `AcquireOperatorToken` and `AcquireTemporaryToken` operation to verify the requesting application. This parameter is also used when consuming the OperatorToken through any of the operations described in section 14.1.6	
		Any string value	Token based on pre-shared security information
client_id (conditional)	String	Used by the `AcquireOperatorToken` operation to identify the requesting application. Used in combination with the `TemporaryToken` serving as secret for authentication. This parameter will be mandatory for `validateOperationToken` in case the user wants to validate **`client_id`** for a specific token. It could be used in combination (for validation) with `scope` parameter.	
		Any string value	Identifier of the requesting application
scope (conditional)	String	Used by the `AcquireOperatorToken` operation to indicate the access privileges being requested for `OperatorToken`. Used in combination with `client_id`. This parameter will be mandatory for `validateOperationToken` in case the user wants to validate **`scope`** for a specific token. It could be used in combination (for validation) with `client_id` parameter.	
		Any string value	Indicates which access privileges are being requested for OperatorToken
msisdn (Conditional)	String	Used by the `VerifyPhoneNumber` operation to compare this value with the one mapped to the token generated during the Authentication process.	
		MSISDN of the subscription in E.164 format.	MSISDN to verify.

Table 114. New parameters for device app authentication

14.1.2 AcquireOperatorToken Operation Configuration Parameters

- Parameter names and presence:
  - `OperatorToken`: Conditional. Operators token to allow authentication for a 3rd party application on the device that may not have the means to acquire token or temporary_token.
  - `OperatorTokenExpiry`: Conditional. Indicates the time the provided OperatorToken expires.
  - `OperatorTokenAuthURL`: Conditional. The URL to representing the endpoint when validating `OperatorToken`
  - `ClientID`: Conditional. ID identifying the requesting application.

The different values for the configuration parameters of the operation `AcquireOperatorToken` are provided in Table 115

"AcquireOperatorToken" configuration parameters	Type	Values	Description
OperatorToken (Conditional)	String	Any string value	This Operator token can be provided by the ECS if the requesting 3rd party application can be authenticated based on `ClientID` and `access_token`. The operator token can be used by the 3rd party application to authenticate the device against the app backend.
OperatorTokenExpiry (Conditional)	Timestamp	ISO 8601 format, of the form YYYY-MM-DDThh:mm:ssTZD	This UTC value provides the expiration time for the Operator token. After the time expiration the Operator token cannot be used for authentication.
OperatorTokenAuthURL (Conditional)	String	URL to validate OperatorToken	URL representing the endpoint to validate the OperatorToken
ClientID (Conditional)	String	Any string value	Identifies the app requesting the OperatorToken

Table 115. Configuration Parameters – AcquireOperatorToken ODSA Operation

14.1.3 AcquireOperatorToken Request Example

Table 116 presents an example for the `AcquireOperatorToken` operation for an ODSA application.

```
GET ? terminal_id = 06170799658&
token = es7w1erXjh%2FEC%2FP8BV44SBmVipg&
terminal_vendor = TVENDOR&
terminal_model = TMODEL&
terminal_sw_version = TSWVERS&
entitlement_version = ENTVERS&
client_id = 087234593407658ß91&
scope = openid%20profile&
app = ap2015&
access_token = ab2d52xaix%2FEC%2FoMNs12Sammctz&
operation = AcquireOperatorToken&
vers = 1 HTTP/1.1

Host: entitlement.telco.net:9014
User-Agent: PRD-TS43 TVENDOR/TMODEL Primary-ODSA/TSWVERS OS-Android/8.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Table 116. Example of an AcquireOperatorToken ODSA Request

14.1.4 AcquireOperatorToken Response Example

Table 117 presents an example for the AcquireOperatorToken response in XML format to a Primary ODSA application. This response provides the TS.43 client with the `OperatorToken` to be used for an app authentication.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS"
       <parm name="version" value="1"/>
       <parm name="validity" value="172800"/>
    </characteristic>

    <characteristic type="TOKEN">
      <parm name="token" value="ASH127AHHA88SF"/>
    </characteristic>

    <characteristic type="APPLICATION">
      <parm name="AppID" value="ap2015"/>
      <parm name="OperatorToken" value="A8daAd8ads7fau34789947kjhsfad;kjfh"/>
      <parm name="OperatorTokenExpiry" value="2019-01-29T13:15:31-08:00"/>
      <parm name="OperatorTokenAuthURL" value="http://verifyurl.example.net"/>
      <parm name="OperationResult" value="1"/>
      <parm name="ClientID" value="684854986221684891O4"/>
    </characteristic>
</wap-provisioningdoc>
```

Table 117. Example of an AcquireOperatorToken Response in XML

14.1.5 Device App authentication with OperatorToken call flow.

Necessary preconditions for this use case:

1. ECS and App Backend exchanged information for OperatorToken encryption.
    - e.g. ECS uses Public Key of App Backend to encrypt OperatorToken. App-Backend can later decrypt OperatorToken with own private Key
2. ECS and App Backend exchanged information for access_token validation.
    - e.g. App Backend uses Public Key of ECS, and forwards information to the app-client on the device. Client can then use this information in the `access_token`

The workflow then follows as described in Figure 64:

3. The 3rd party App requests an Operator Token from the TS.43 client of the device
4. The TS.43 client initiates the EAP-AKA authentication procedure with the ECS, using `app_ID` ap2015.
5. Device and ECS perform EAP-AKA authentication as described in section 2.8.1.
6. The TS.43 requests a TemporaryToken, using the EAP-AKA token and the `access_token` of the 3rd party app. The `operation_target` should be AcquireOperatorToken.
7. The ECS validates the request including the identifiers and the AuthToken. The `access_token` is validated with the information shared between the ECS and the app_backend. Optionally the ECS can also verify the access_token with the 3rd party app backend. When successful, ECS creates the temporary_token.

8. ECS sends the AcquireTemporaryToken response including TemporaryToken, `TemporaryTokenExpiry` and the `OperationTargets` = “AcquireOperatorToken”. The client stores the temporary_token in a secured space not accessible to 3rd party apps.
9. Client uses the temporary_token to acquire the OperatorToken. In the AcquireOperatorToken Request the client also provides the client_id, which uniquely identifies the app.
   If the client & ECS support encrypting all information in one token, steps 6, 7 and 8 are optional here.
   If steps 6, 7 and 8 are skipped the client should send the access_token in this step.
10. ECS validates the temporary_token together with the client_id. e.g. the ECS could use OAuth with client_id and temporary_token as secret. If successful, the ECS generates the `OperatorToken`
11. ECS sends AcquireOperatorToken response, including OperatorToken, OperatorTokenExpiry, OperatorTokenAuthURL & ClientID
12. TS.43 Client forwards the OperatorToken to the 3rd Party App
13. The 3rd Party App can use OperatorToken to authenticate at its own Backend Service. By using OperatorToken, the device is authenticated by the MNO based on the inserted SIM-Card

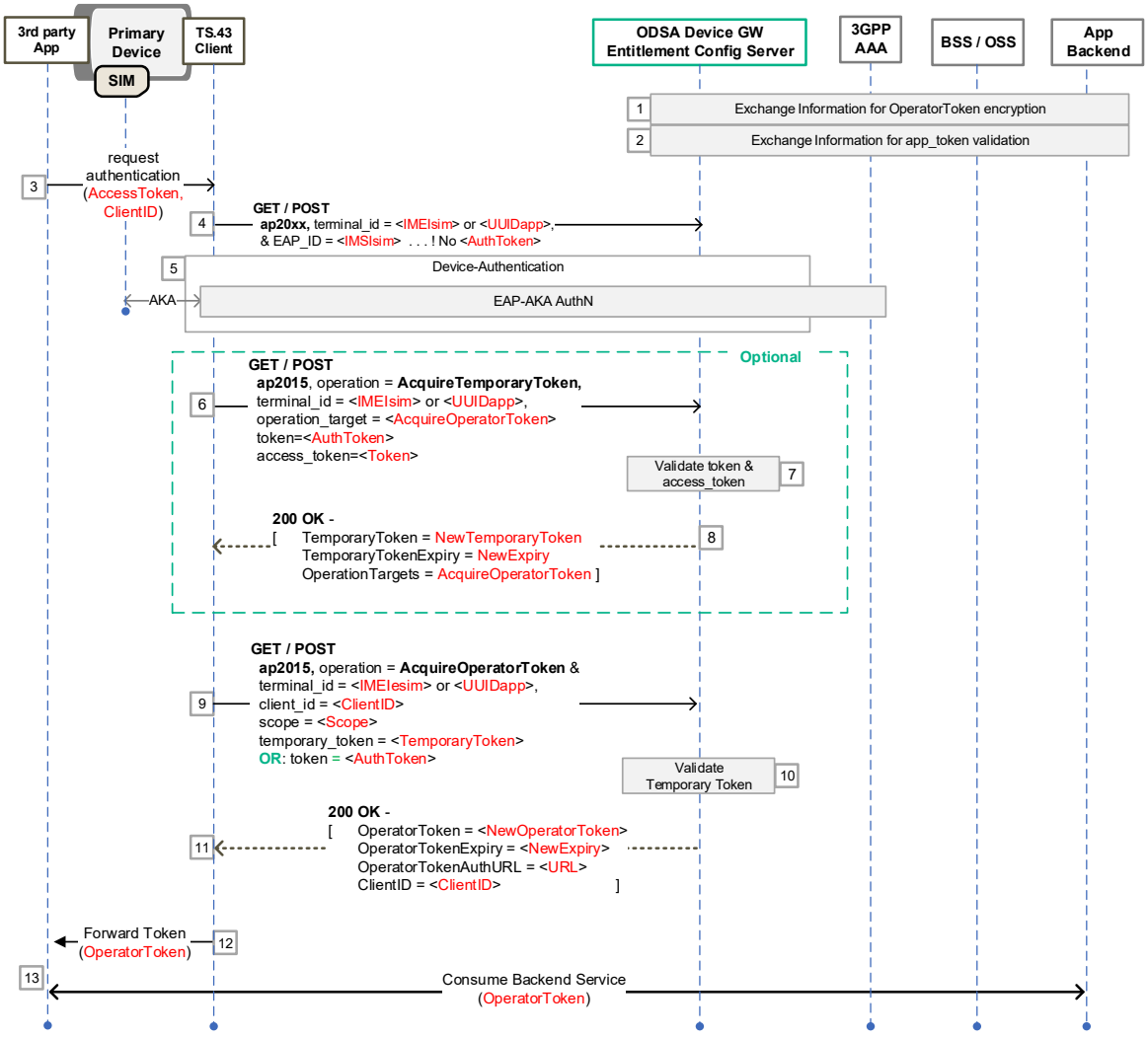

Figure 64. App Authentication using OperatorToken.

14.1.6 Operator Token Consumption

An **External Entity** (e.g. Application Server, etc.), may use the OperatorToken implementing new operations that will be specific for **ap2015**. These new operations are identified in Table 118.

It's important to note that external entity could not be a terminal but a server, but, even so, in the request there will be some parameters referring to terminal_* present on the requests as part of the RCC.14 standard. For these mandatory parameters, it is recommended to use dummy values.

"Operator Token" operations	Section	Description
ValidateOperatorToken	14.1.6.1	Validates the operator token for a specific **`client_id`** and/or **`scope`**. This operation requires as part of the request, at least, one of the following parameters to be checked: client_id, scope.
GetSubscriberDeviceInfo	14.1.6.2	Provides information related to the subscriber device that acquired the operator token.
VerifyPhoneNumber	14.1.7	Verifies if the MSISDN provided in the request maps to the MSISDN from terminal_id belonging the token for Authentication.

Table 118. Operations available for Operator Token usage

Operations in Table 118 needs to be mapped to one or more scopes for validation. This scope definition is out of scope of TS.43 and should be the ECS (as it is the system generating the operator token) the one taking care of this mapping.

Using operations like the ones defined in Table 118 is similar, and the flow will follow the example as described in Figure 65, where:

1. The External Entity makes a request using the `operator_token` and for a specific `operation`.
2. ECS checks the validity of the `operator_token`. Validation could also require crosschecking with `requestor_id`.
3. Optional. Depending on the operation, ECS could require interacting with backend systems.
4. As a result, ECS will send the response containing the response parameters specific to the `operation`.

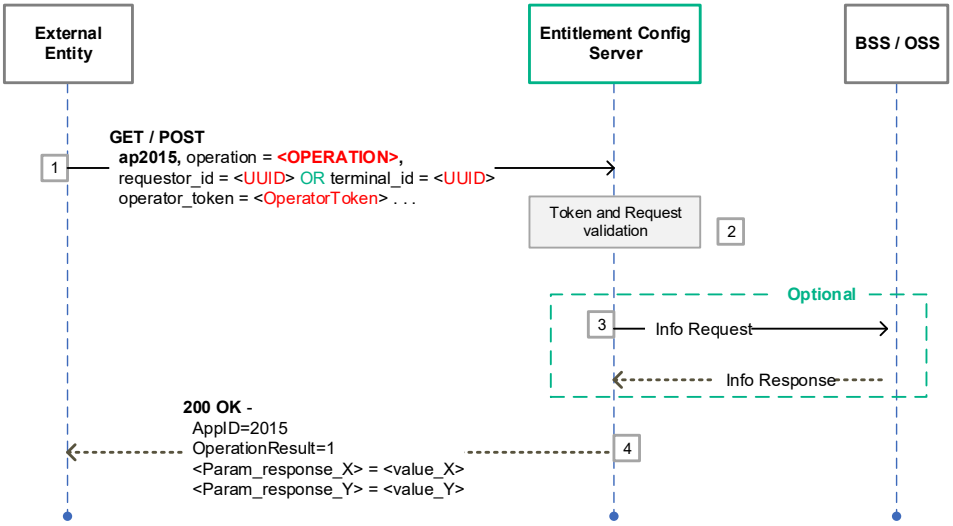

Figure 65. Example for Operator Token Usage Flow

Application requests using operator token mainly differs in the operation parameter. Table 119 shows a generic example which could be applicable for any request.

```
GET ? requestor_id = 06170799658&
terminal_vendor = TVENDOR&
terminal_model = TMODEL&
terminal_sw_version = TSWVERS&
entitlement_version = ENTVERS&
app = ap2015&
operator_token = <OPERATOR_TOKEN>&
operation = <OPERATION>&
scope= <SCOPE>&
access_token = <ACCESS_TOKEN>&   // Optional
vers = 1 HTTP/1.1

Host: entitlement.telco.net:9014
User-Agent: PRD-TS43 TVENDOR/TMODEL Primary-ODSA/TSWVERS OS-Android/8.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Table 119. Generic operation request for specific Operator Token Usage

The same approach is used for the responses where the main difference between each of the operations will defer in the response parameters. Table 120 shows a generic example where the response contains two parameters (`<Param_response_X>` and `<Param_response_Y>`).

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS"
       <parm name="version" value="1"/>
       <parm name="validity" value="172800"/>
    </characteristic>

    <characteristic type="APPLICATION">
      <parm name="AppID" value="ap2015"/>
      <parm name="OperationResult" value="1"/>
      <parm name="<Param_response_X>" value="<Param_response_X_value>"/>
      <parm name="<Param_response_Y>" value="<Param_response_Y_value>"/>
    </characteristic>

</wap-provisioningdoc>
```

Table 120. Generic operation request for specific Operator Token Usage

IMPORTANT.- Note that Operation Token Usage responses (Table 119) **do not contain a token** for Fast Authentication.

14.1.6.1 Operator Token Validation

Parameters in the response are described in Table 121.

ValidateOperatorToken response parameters	Type	Values	Description
OperatorTokenValidity	Integer	Indicates if the parameters (`client_id` and/or `scope`) are valid ones for the specific operator token in the request.	
		0 – NOT VALID	The `operator token` provided is not a valid one
		1 – VALID	The `operator token` provided is valid one
OperatorTokenValidatedParams (Optional)	String	Comma-separated list with parameters validated.	List the parameters that have been validated for the `operator_token`.

Table 121. Response parameters for ValidateOperatorToken operation

Table 122 shows an example of a request for `ValidateOperatorToken`, validating the OperatorToken for a specific `scope` and `client_id`.

```
GET ? requestor_id = 06170799658&
terminal_vendor = TVENDOR&
terminal_model = TMODEL&
terminal_sw_version = TSWVERS&
entitlement_version = ENTVERS&
app = ap2015&
operator_token = ab2d52xaix%2FEC%2FoMNs12Sammctz&
operation = ValidateOperatorToken&
scope= "scope1"&
client_id= "25625441&
access_token = 32487234987238974&   // Optional
vers = 1 HTTP/1.1

Host: entitlement.telco.net:9014
User-Agent: PRD-TS43 TVENDOR/TMODEL Primary-ODSA/TSWVERS OS-Android/8.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Table 122. ValidateOperatorToken Request example

Table 123 shows a response example for the previous (`ValidateOperatorToken`) request (Table 122).

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS"
       <parm name="version" value="1"/>
       <parm name="validity" value="172800"/>
    </characteristic>

    <characteristic type="APPLICATION">
      <parm name="AppID" value="ap2015"/>
      <parm name="OperationResult" value="1"/>
      <parm name="OperatorTokenValidity" value="1"/>
      <parm name="OperatorTokenValidatedParams" value="scope,client_id"/>
    </characteristic>

</wap-provisioningdoc>
```

Table 123. ValidateOperatorToken Response example

14.1.6.2 Subscriber Device Information

Parameters in the response are described in Table 124.

GetSubscriberDeviceInfo response parameters	Type	Values	Description
MSISDN	String	Any string value	E.164 formatted phone number. It is possible to provide the base64 encoding of the value by preceding it with `encodedValue=`
IMSI (Optional)	String	A 15 digits (max) string	International Mobile Subscriber Identity as per ITU E.212 or 3GPP TS 23.003 standards.

Table 124. Response parameters for GetSubscriberDeviceInfo operation

Table 125 shows an example of a request for `GetSusbcriberDeviceInfo`.

```
GET ? requestor_id = 06170799658&
terminal_vendor = TVENDOR&
terminal_model = TMODEL&
terminal_sw_version = TSWVERS&
entitlement_version = ENTVERS&
app = ap2015&
operator_token = ab2d52xaix%2FEC%2FoMNs12Sammctz&
operation = GetSubscriberDeviceInfo&
access_token = 32487234987238974&   // Optional
vers = 1 HTTP/1.1

Host: entitlement.telco.net:9014
User-Agent: PRD-TS43 TVENDOR/TMODEL Primary-ODSA/TSWVERS OS-Android/8.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Table 125. GetSubscriberDeviceInfo Request example

Table 126 shows a response example for the previous (`GetSubscriberDeviceInfo`) request (Table 125).

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS"
       <parm name="version" value="1"/>
       <parm name="validity" value="172800"/>
    </characteristic>

    <characteristic type="APPLICATION">
      <parm name="AppID" value="ap2015"/>
      <parm name="OperationResult" value="1"/>
      <parm name="MSISDN" value="+34616210000"/>
      <parm name="IMSI" value="214990011223344"/>
    </characteristic>

</wap-provisioningdoc>
```

Table 126. GetSubscriberDeviceInfo Response example

14.1.7 Phone Number Verification

Parameters in the response are described in Table 127

VerifyPhoneNumber response parameters	Type	Values	Description
PhoneNumberVerification	Integer	Indicates the result of the Phone Number verification	
		0 – FAILURE	MSISDNs don't match
		1 – SUCCESS	MSISDNs match

VerifyPhoneNumber response parameters	Type	Values	Description
msisdn (Optional)	String	This parameter could be present when SUCCESS. If present, it indicates the MSISDN (the one from the request) that has been verified successfully.	

Table 127 Response parameters for VerifyPhoneNumber operation

```
GET ? requestor_id = 06170799658&
terminal_vendor = TVENDOR&
terminal_model = TMODEL&
terminal_sw_version = TSWVERS&
entitlement_version = ENTVERS&
app = ap2015&
operator_token = ab2d52xaix%2FEC%2FoMNs12Sammctz&
operation = VerifyPhoneNumber&
msisdn = "+34616210000"
access_token = 32487234987238974&   // Optional
vers = 1 HTTP/1.1

Host: entitlement.telco.net:9014
User-Agent: PRD-TS43 TVENDOR/TMODEL Primary-ODSA/TSWVERS OS-Android/8.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Table 128 VerifyPhoneNumber Request example

Table 129 shows a response example for the previous (`VerifyPhoneNumber`) request (Table 128).

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS"
       <parm name="version" value="1"/>
       <parm name="validity" value="172800"/>
    </characteristic>

    <characteristic type="APPLICATION">
      <parm name="AppID" value="ap2015"/>
      <parm name="OperationResult" value="1"/>
      <parm name="PhoneNumberVerification" value="1"/>
      <parm name="msisdn" value="+34616210000"/>
    </characteristic>

</wap-provisioningdoc>
```

Table 129. VerifyPhoneNumber Response example

14.2 App token use case

For some use cases, like accessing operator special network capabilities, there might be situations where the device needs to be aware of application information managed by the network operator. For this case, devices require additional parameters in the HTTP requests, outside of the ones described in 2.3, 6.2 and 14.1.1, which are described in the following sections presenting the new parameters and their associated operations.

14.2.1 App Token consumption

For the consumption of the App token, the new operation and parameters described in Table 130 are needed.

App token may be used only once, and once it is consumed by a device invoking the Get3ppAppInfo operation, it shall become useless (further operations shall return an error). Additionally, it should have associated a short expiration time (minutes range).

New GET parameters app information	Type	Values	Description
operation	String	`Get3PAppInfo`	Indicates the operation requested by the TS.43 client
App_token	String	Used by the `Get3PAppInfo` operation to verify the application information managed by the operator.	
		Any string value	Token based on pre-shared security information. It can be generated as per section 2.8.3. Note that App_token is going to be used by the ECS to identify the 3rd party application backend (server) who requested this token.

Table 130. Request Parameters – Get3PAppInfo ODSA Operation

In Table 131 there is also a description about the parameters and presence needed for the Get3PAppInfo response.

- Parameter names and presence:
  - `3PAppNameAnon`: Mandatory. The application name anonymized corresponding to the operator managed application information. Real application name cannot be obtained by the operator from this parameter because only 3rd party application server should be able to obtain the real application name.

"Get3PAppInfo" configuration parameters	Type	Values	Description
3PAppNameAnon	String	Any string value	3rd party application name anonymized by the 3rd party application server.

Table 131. Configuration Parameters – Get3PAppInfo ODSA Operation

Table 132 presents an example for the `Get3PAppInfo` operation for an ODSA application.

```
GET ? terminal_id = 06170799658&
token = es7w1erXjh%2FEC%2FP8BV44SBmVipg&
terminal_vendor = TVENDOR&
terminal_model = TMODEL&
terminal_sw_version = TSWVERS&
entitlement_version = ENTVERS&
app = ap2015&
App_token = ab2d52xaix%2FEC%2FoMNs12Sammctz&
operation = Get3PAppInfo&
vers = 1 HTTP/1.1

Host: entitlement.telco.net:9014
User-Agent: PRD-TS43 TVENDOR/TMODEL Primary-ODSA/TSWVERS OS-Android/8.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Table 132. Example of an Get3PAppInfo ODSA Request

Table 133 presents an example for the `Get3PAppInfo` response in XML format to a Primary ODSA application. This response provides the TS.43 client with the 3rd party application info managed by the operator.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS"
       <parm name="version" value="1"/>
       <parm name="validity" value="172800"/>
    </characteristic>

    <characteristic type="APPLICATION">
      <parm name="AppID" value="ap2015"/>
      <parm name="3PAppNameAnon" value="ZacaPWVIH44fCZ3D"/>
</characteristic>
</wap-provisioningdoc>
```

Table 133. Example of a Get3PAppInfo Response in XML

14.2.2 Discovering 3rd party application information callflow

- Necessary preconditions for this use case:

1. ECS and App Backend exchange information for App_token generation. App Backend delivers App token to 3rd party application installed in the device.

The workflow then follows as described in Figure 66:

2. The 3rd party App provides the App token to the TS.43 client of the device to request 3rd party application information.
3. The TS.43 client initiates the EAP-AKA authentication procedure with the ECS, using `app_ID` ap2015. Device and ECS perform EAP-AKA authentication as described in section 2.8.1.
4. Client requests 3rd party application information by using `Get3PAppInfo` operation providing `App_token` as input attribute.
5. The `App_token` is used by the ECS to obtain the application information associated to the application backend (server).

6. ECS sends `Get3PAppInfo` response, including `3pAppNameAnon`.
7. 3rd party application information managed by the operator can be used in the device.

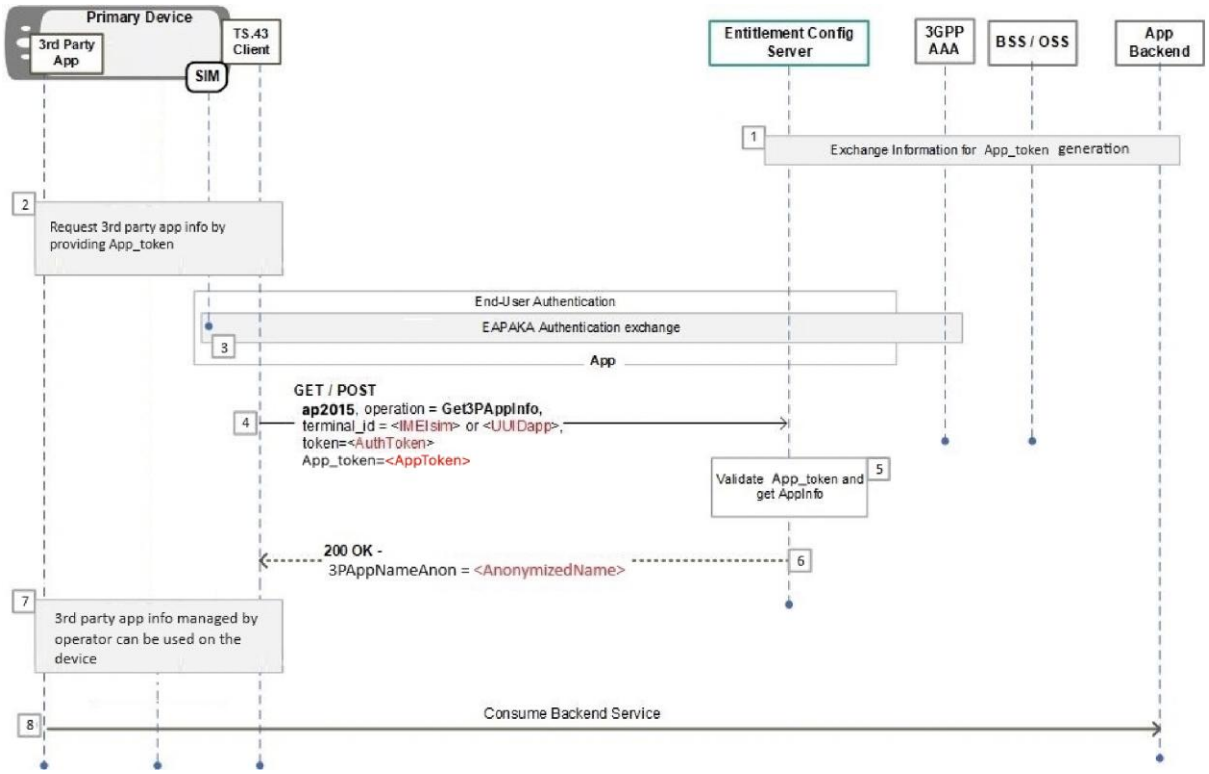

Figure 66. Obtaining app information managed by the operator by using App token

15 SatMode Entitlement and Provisioning

The following describes the use case of Satellite Mode (SatMode) Entitlement and Provisioning, where the application requires entitlement and PLMN List to attach to the network. It is identified by the appID “appID=**ap2016**”. The client should only fetch SatMode configurations if the deviceType and OS Version is capable. The full flowchart diagrams can be found in section 15.4.

The following sections describe the different configuration parameters associated with the SatMode entitlement as well as the expected behaviour of the client based on the entitlement configuration document received by the client.

15.1 SatMode Entitlement Parameters

Parameters for the SatMode entitlement provide the overall status of the service to the client, as well as the different sub-status associated with the activation procedure of the service.

The SatMode entitlement uses the `EntitlementStatus` parameter, and also includes information associated with the web views presented to users by the SatMode client during activation and management of the service, as described in the following.

15.1.1 SatMode Entitlement Status

- Parameter Name: `EntitlementStatus`
- Presence: Mandatory

This parameter indicates the overall status of the SatMode entitlement, stating if the service can be offered on the device, and if it can be activated or not by the end-user.

The different values for the SatMode entitlement status are provided in Table 134.

SatMode Entitlement parameter	Type	Values	Description
EntitlementStatus (Mandatory)	Integer	0 - DISABLED	SatMode allowed, but not yet provisioned and activated on the network. Device is redirected to webview using ServiceFlow_URL
		1 - ENABLED	SatMode service allowed, provisioned and activated on the network
		2 - INCOMPATIBLE	SatMode cannot be offered for network or device
		3 - PROVISIONING	SatMode is being provisioned on the network

Table 134. Entitlement Parameter - SatMode Overall Status

15.1.2 SatMode Entitlement Request Example

Table 135 presents an example for the entitlement request for a SatMode application.

```
GET ? terminal_id = 013787006099922&
token = es7w1erXjh%2FEC%2FP8BV44SBmVipg&
terminal_vendor = TVENDOR&
terminal_model = TMODEL&
terminal_sw_version = TSWVERS&
entitlement_version = ENTVERS&
app = ap2016&
vers = 1 HTTP/1.1

Host: entitlement.telco.net:9014
User-Agent: PRD-TS43 TVENDOR/TMODEL SatMode/TSWVERS OS-Android/8.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Table 135. Example of a SatMode Entitlement Request

15.1.3 SatMode Entitlement Response Example

Table 136 presents an example for the entitlement response in XML format for a SatMode application.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS">
         <parm name="version" value="X"/>
         <parm name="validity" value="Y"/>
    </characteristic>
    <characteristic type="TOKEN">
         <parm name="token" value="U"/>
    </characteristic>
    <characteristic type="APPLICATION">
        <parm name="AppID" value="ap2016"/>
        <parm name="EntitlementStatus" value="0"/>
        <parm name="ServiceFlow_URL"
value="https://www.MNO.org/serviceActivation"/>
        <parm name="ServiceFlow_UserData"value="UserData"/>
    </characteristic>
</wap-provisioningdoc>
```

Table 136. Example of a SatMode Entitlement Response in XML format

Table 137 presents an example for the entitlement response in JSON format for a SatMode application.

```
{
    "Vers" : {
        "version" : "1",
        "validity" : "172800"
    },     "Token" : {
        "token" : "ASH127AHHA88SF"
    },
    "ap2016” : {
        "EntitlementStatus" : "0",
        "ServiceFlow_URL" : "https://www.MNO.org/serviceActivation",
        "ServiceFlow_UserData" : "UserData"
    }
}
```

Table 137. Example of a SatMode Entitlement Response

15.1.4 SatMode Activation Web Views Parameters

- Parameter Names: `ServiceFlow_URL` and `ServiceFlow_UserData`
- Presence: Conditional

During the activation procedure of the SatMode service, end-users can be presented with web views specific to the Service Provider. SatMode web views allow end-users to change user-specific attributes of the SatMode service, like the acceptance of the service's Terms and Conditions (T&C) and list plan selection for end-user to choose from.

The entitlement parameters associated with the SatMode service's web views are described in Table 138.

SatMode Entitlement parameter	Type	Values	Description
ServiceFlow_URL (Conditional)	String	URL to a Service Provider site or portal	The URL of web views to be used by client to present the user with SatMode service activation and service management options, which may include agreeing to the T&C of the SatMode service.
ServiceFlow_UserData (Conditional)	String	Parameters or content to insert when invoking URL provided in the `ServiceFlow_URL` parameter	User data associated with the HTTP web request towards the ServiceFlow URL. It can contain user-specific attributes to ease the flow of SatMode service activation and management. See below for details on the content.
ServiceFlow_ContentsType (Conditional)	String	Specifies content and HTTP method to use when reaching out to the web server specified in `ServiceFlow_URL`.	
		Not present	Method to `ServiceFlow_URL` is HTTP GET request with query parameters from `ServiceFlow_UserData`.
		Json	Method to `ServiceFlow_URL` is HTTP POST request with JSON content from `ServiceFlow_UserData`.
		XML	Method to `ServiceFlow_URL` is HTTP POST request with XML content from `ServiceFlow_UserData`.

Table 138. Entitlement Parameters - SatMode Web Views Information

The content of the `ServiceFlow_UserData` parameter is defined by the requirements of the Service Provider's SatMode web views. In a typical case, the web view is presented when SatMode service is activated by the end-user. At such time the client connects the user to the `ServiceFlow_URL` and includes the `ServiceFlow_UserData` in the HTTP web request.

In order to improve user experience, this parameter should include user and service-specific information that would allow the SatMode's web views to identify the requestor and be aware of the latest SatMode entitlement status values.

An example of the `ServiceFlow_UserData` string is:

```
"imsi=XXXXXXXXX&amp;msisdn=XXXXXXXX&amp;tnc=X&amp;prov=X&
amp;device_id=XXXXXXXX&amp;entitlement_name=SatMode"
```

This example contains elements associated with the device and user identities as well as service-related information like the current T&C and provisioning status of the SatMode service. Note the use of "`&`" is required to allow the '&' character to be used in a string value within an XML document.

15.1.5 SatMode Message for Incompatible Status

- Parameter Name: `MessageForIncompatible`
- Presence: Mandatory

When the status for the SatMode entitlement is INCOMPATIBLE (see Table 134) and the end-user tries to activate SatMode, the client should show a message to the end-user indicating why activation was refused.

This entitlement parameter provides the content of that message, as decided by the Service Provider. Table 139 describes this SatMode entitlement parameter.

SatMode Entitlement parameter	Type	Description
MessageForIncompatible (Mandatory)	String	A message to be displayed to the end-user when activation fails due to an incompatible SatMode Entitlement Status

Table 139. Entitlement Parameter - SatMode Message for Incompatible Status

15.2 SatMode Config Parameters

`PLMNAllowed` parameter is conditional upon EntitlementStatus = 1-ENABLED and client shall use this to connect to the appropriate PLMN.

`PLMNBarred` parameter is optional and can be present in any of the SatMode entitlement status.

`PLMNAllowedDuringDisasters` parameter is optional and applies when EntitlementStatus = 1-ENABLED. The client shall use this to connect to the appropriate PLMN only when the device determines that it is experiencing a disaster situation. If present, the client will ignore the `PLMNAllowed` parameter when a disaster situation occurs.

`PLMNBarredDuringDisasters` parameter is optional and applies when EntitlementStatus = 1-ENABLED. The client shall use this parameter only when the device determines that it is experiencing a disaster situation. If present the client will ignore the `PLMNBarred` parameter when a disaster situation occurs.

The parameter structures are as below:

SatMode Config parameter	Type	Values	Description
PLMNAllowed (Conditional)	LIST of Objects	multi-parameter value - see Table 141 for details	Top level, list of allowed PLMNs where the service can be used.
PLMNBarred (Optional)	LIST of Objects	multi-parameter value - see Table 141 for details	Top level, list of barred PLMNs where the service can't be used.
PLMNAllowedDuringDisasters (Optional)	LIST of Objects	multi-parameter value - see Table 141 for details	Top level, list of allowed PLMNs where the service can be used.
PLMNBarredDuringDisasters (Optional)	LIST of Objects	multi-parameter value - see Table 141 for details	Top level, list of barred PLMNs where the service can't be used.

Table 140. SatModeConfig- Parameters

PLMNAllowed / PLMNBarred Parameter	Type	Values	Description
PLMN (Conditional)	String	PLMN ID	allowed PLMN-ID where the service can be used or is barred.
DataPlanType (Optional)	String	`Metered`	The data plan is of the metered type
		`Unmetered`	The data plan is of the un-metered type
AllowedServicesInfo (Optional)	Array	Array of AllowedServices – See Table 142 for details	Array of allowed services, in addition to carrier messaging, that are allowed over satellite Note: If Parameter "AllowedServicesInfo" is not present, it is assumed that carrier messaging is supported by default

Table 141. PLMNAllowed & PLMNBarred- Parameters

AllowedServices	Type	Values	Description
ServiceType	String	`data`	Data is supported
		`voice`	Voice is supported

AllowedServices	Type	Values	Description
ServicePolicy	String	`Constrained`	The data rate available for the satellite connection is bandwidth limited
		`Unconstrained`	The data rate available for the satellite connection is not bandwidth limited

Table 142. AllowedServices- Parameters

15.2.1 SatMode Config Request example

Table 143 presents an example for the SatModeConfig use case.

```
GET ? terminal_id = 013787006099922&
token = es7w1erXjh%2FEC%2FP8BV44SBmVipg&
terminal_vendor = TVENDOR&
terminal_model = TMODEL&
terminal_sw_version = TSWVERS&
entitlement_version = ENTVERS&
app = ap2016&
vers = 1 HTTP/1.1

Host: entitlement.telco.net:9014
User-Agent: PRD-TS43 TVENDOR/TMODEL SatMode/TSWVERS OS-Android/8.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Table 143. Example of a SatModeConfig Request

15.2.2 SatModeConfig Response Example

Table 144 presents an example for the SatModeConfig use case response in XML format.

```
<?xml version="1.0"?>
<wap-provisioningdoc version="1.1">
    <characteristic type="VERS">
        <parm name="version" value="1"/>
        <parm name="validity" value="172800"/>
    </characteristic>
    <characteristic type="TOKEN">
        <parm name="token" value="ASH127AHHA88SF"/>
    </characteristic>
    <characteristic type="APPLICATION">
        <parm name="AppID" value="ap2016"/>
        <parm name="EntitlementStatus" value="1"/>
        <characteristic type="PLMNAllowed">
            <parm name="PLMN" value="31026"/>
            <parm name="DataPlanType" value="metered"/>
        </characteristic>
        <characteristic type="PLMNAllowed">
            <parm name="PLMN" value="302820"/>
            <parm name="DataPlanType" value="unmetered"/>
            <characteristic type="AllowedServicesInfo">
                <characteristic type="AllowedServices">
                    <parm name="ServiceType" value="data"/>
                    <parm name="ServicePolicy" value="constrained"/>
                </characteristic>
                <characteristic type="AllowedServices">
                     <parm name="ServiceType" value="voice"/>
                           <parm name="ServicePolicy" value="unconstrained"/>
      </characteristic>
            </characteristic>

        </characteristic>
        <characteristic type="PLMNBarred">
            <parm name="PLMN" value="31017"/>
        </characteristic>
        <characteristic type="PLMNBarred">
            <parm name="PLMN" value="302020"/>
        </characteristic>
    </characteristic>
</wap-provisioningdoc>
```

Table 144. Example of a SatModeConfig Response

Table 145 presents an example for the SatModeConfig use case response in JSON format.

```
{
    "Vers": {
        "version": "1",
        "validity": "172800"
    },
    "Token": {
        "token": "ASH127AHHA88SF"
    },
    "ap2016": {
        "PLMNAllowed": [
            {
            "PLMN": "31026",
            "DataPlanType": "unmetered"},
            {
            "PLMN": "302820",
            "DataPlanType": "metered"},
            "AllowedServicesInfo: [{
                "AllowedServices: {
                    "ServiceType": "data",
                    "ServicePolicy": "constrained"},
                },{
                "AllowedServices: {
                    "ServiceType": "voice",
                    "ServicePolicy": "unconstrained"}
                }]
}
        ],
        "PLMNBarred": [
            {"PLMN": "31017"},
            {"PLMN": "302020"}
        ]
    }
}
```

Table 145. Example of a SatModeConfig Response

15.3 SatMode Config retrieval frequency

Client may refresh the service Config at a regular interval or one of the scenarios listed below (list is not exhaustive):

- At a pre-defined regular interval, like 1 day or 7 days
- Upon Device power cycle.
- SIM Swap.
- Airplane mode disabled.
- Notification received from the Service Provider.
- Software version update

15.4 SatMode Client Considerations around Web View Callbacks

During the activation procedure of the SatMode service, end-users can be presented with web views specific to the Service Provider (hosted by a SatMode portal web server). To support this feature, the SatMode entitlement parameters **`ServiceFlow_URL`** and **`ServiceFlow_UserData`** associated with the invocation of SatMode service's web views by the SatMode client are defined in section 15.1.4.

At the completion of the web service flow by the SatMode portal web server, the web page shall invoke a specific JavaScript (JS) callback function associated with the SatMode client.

The callback functions shall provide the overall state of the web flow to the SatMode client and indicate that the SatMode web view on the device needs to be closed.

The object associated with the callback functions is **`SatModeWebServiceFlow`** and two different callback functions are defined to reflect the state of the web logic.

15.4.1 entitlementChanged() Callback function

The **`entitlementChanged()`** callback function indicates that the SatMode service flow ended properly between the device and SatMode portal web server.

The web view to the end-user should be closed and the SatMode client shall make a request for the latest SatMode entitlement configuration status, via the proper TS.43 entitlement configuration request.

The following call flow presents how the **`entitlementChanged()`** callback function fits into the typical steps involved with SatMode entitlement configuration. At the end of the SatMode service flow the callback function (step 6) is invoked by the web server and the SatMode client acts accordingly by requesting for the latest SatMode entitlement configuration.

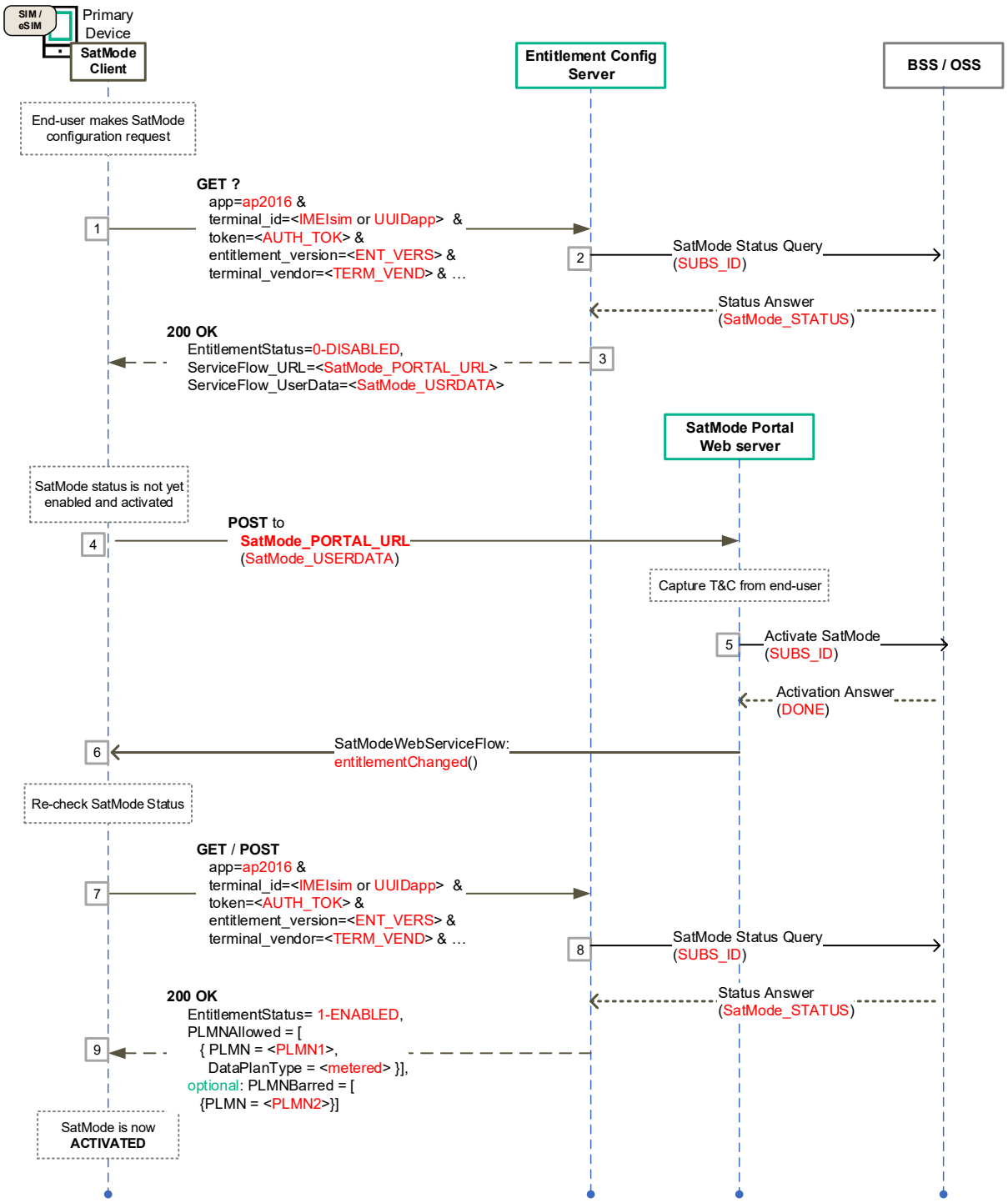

Figure 67. SatMode Entitlement Configuration Flow with entitlementChanged() Callback

15.4.2 dismissFlow() callback function

The **`dismissFlow()`** callback function indicates that the SatMode service flow ends prematurely, either caused by user action (DISMISS button for example) or by an error in the web sheet logic or from the network side.

As a result of the dismissal of the service flow, the SatMode entitlement status has not been updated by the SatMode portal.

The web view to the end-user should be closed and the SatMode client should not make a request for the latest SatMode entitlement configuration status.

The call flow in Figure 68 presents how the **`dismissFlow()`** callback function fits into the typical steps involved with SatMode Entitlement Configuration. Due to an error or user action the callback function (step 6) is invoked by the web server and the SatMode client acts accordingly.

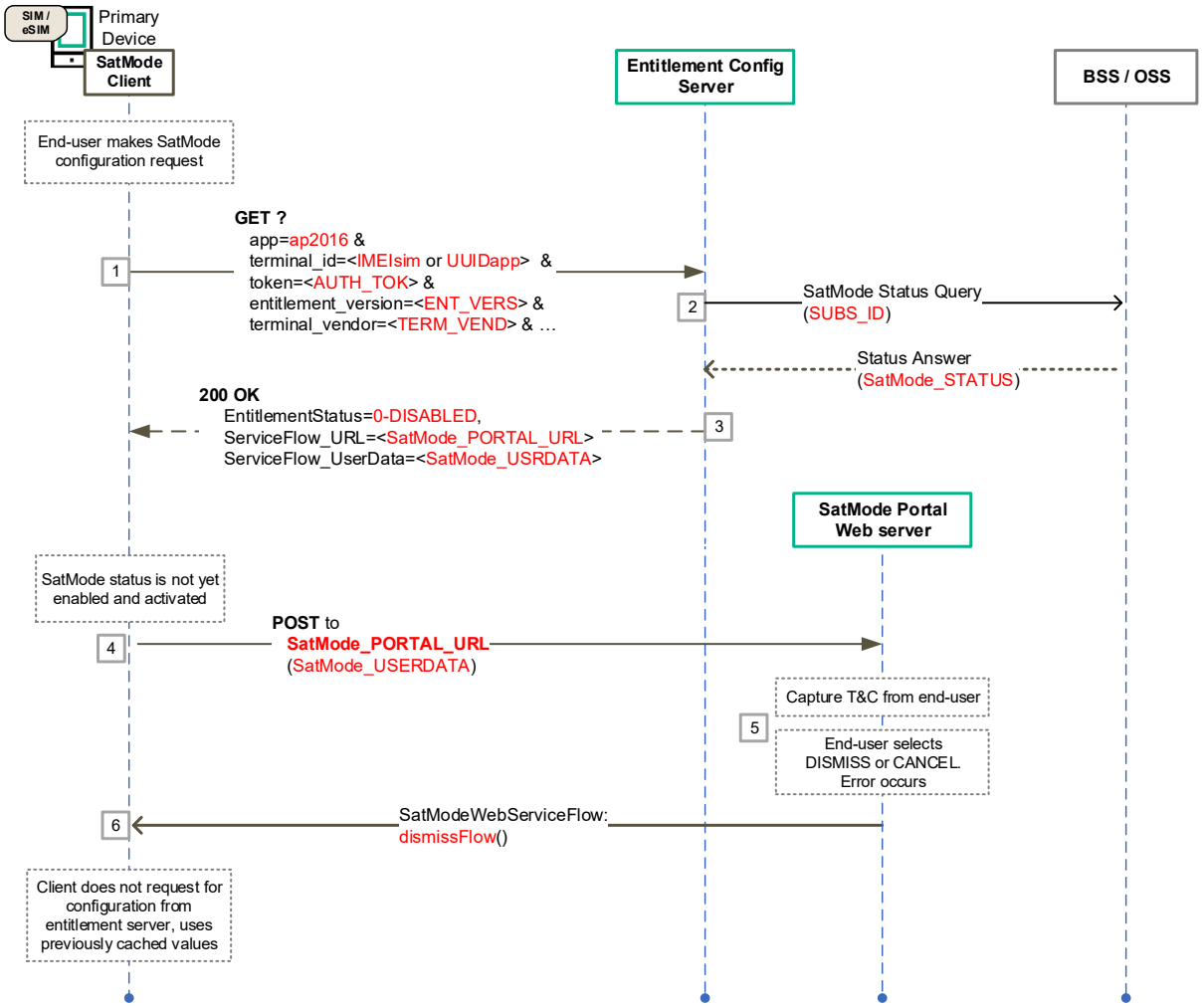

Figure 68. SatMode Entitlement Configuration Flow with dismissFlow() Callback

Annex A Feature mapping

A.1 Feature and procedure lists

This section is dedicated to help a new reader finding what is the status of the different operations & parameters among features.

The features considered are the entitlement configuration use-cases identified by their appID: **VoWiFi** (ap2004), **Voice over Cellular** (ap2003), **SMSoIP** (ap2005), **ODSA for Companion** (ap2006), **ODSA for Primary** (ap2009), **Data Plan Information** (ap2010), **ODSA for Server Initiated Request** (ap2011), **Direct Carrier Billing** (ap2012), **Private User Identity** (ap2013), **Device and User Information** (ap2014), **Device App Authentication with OperatorToken** (ap2015) and **SatMode Entitlement** (ap2016).

The procedures considered are:

- authenticate the Subscriber Identity
- check the compliance of the device & user subscription with the requested service.
- get the entitlement configuration document.
- manage the user subscription.
- get the user consent.
- update the configuration document from the network.

For each feature, the procedures status may be: Mandatory (**M**), Optional (**O**), Conditional (**C**) or Not Applicable (**N/A**).

The procedures are detailed in operations.

In each case, the mapping references the related section for the Service Provider’s Entitlement Configuration Server and the client.

A.2 VoWiFi feature

Procedure	Operation	Entitlement Client		SP Entitlement Server	
Authenticate the Subscriber Identity	Embedded EAP-AKA authentication	M	2.8.1	M	2.8.1
	Fast authentication	M	2.8.5	M	2.8.5
Get the entitlement configuration document	HTTP GET	M M	2.8.7 3.2	M	3.1
Get the user consent	Display webviews	O	3.4	M	3.1.4
Update the entitlement configuration from network	Push notification	O	2.6.2	O	2.6.2

Table 146. Features & operations mapping for VoWiFi.

A.3 Voice over Cellular feature

Procedure	Operation	Entitlement Client		SP Entitlement Server	
Authenticate the Subscriber Identity	Embedded EAP-AKA authentication	M	2.8.1	M	2.8.1
	Fast authentication	M	2.8.5	M	2.8.5
Get the entitlement configuration document	HTTP GET	M	2.8.7	M	4.1
Update the entitlement configuration from network	Push notification	O	2.6.2	O	2.6.2

Table 147. Features & operations mapping for Voice over Cellular

A.4 SMSoIP feature

Procedure	Operation	Entitlement Client		SP Entitlement Server	
Authenticate the Subscriber Identity	Embedded EAP-AKA authentication	M	2.8.1	M	2.8.1
	Fast authentication	M	2.8.5	M	2.8.5
Get the entitlement configuration document	HTTP GET	M	5.2	M	5.1
Update the entitlement configuration from network	Push notification	O	2.6.2	O	2.6.2

Table 148. Features & operations mapping for SMSoIP.

A.5 Companion ODSA feature

Procedure	Operation	Entitlement Client		SP Entitlement Server	
Authenticate the Subscriber Identity	Embedded EAP-AKA authentication	O	2.8.1	O	2.8.1
	Fast authentication	M	2.8.5	M	2.8.5
	OAuth2.0/OIDC authentication	O	2.8.2	O	2.8.2
Check the compliance of the device and user subscription with the requested service	CheckEligibility	M	6.2	M M	6.5.1 6.5.2
Get the entitlement configuration document	AcquireConfiguration	M	6.2	M M	6.5.1 6.5.5
Get user consent	Display WebView	O	6.7	O	6.7
Manage subscription	ManageSubscription,		6.2	M M	6.5.1 6.4.2
	Display WebView	O	6.7	O	6.7
Change the service status from client	ManageService	O	6.2	M M	6.5.1 6.5.4
Update the entitlement configuration from network	Push notification	C1	2.6	C1	2.6
	Polling	C1	7.3		

Table 149. Features & operations mapping for Companion ODSA

C1: IF *Push notification* IS NOT SUPPORTED THEN *POLLING* IS *M*

A.6 Primary ODSA feature

Procedure	Operation	Entitlement Client		SP Entitlement Server	
Authenticate the Subscriber Identity	Embedded EAP-AKA authentication	O	2.8.1	O	2.8.1
	Fast authentication	M	2.8.5	M	2.8.5
	OAuth2.0/OIDC authentication	O	2.8.2	O	2.8.2
	Use a temporary token for specific operation (AcquireTemporaryToken)	O O	6.1, 6.2 6.5.7	O O	6.2 6.5.7
Check the compliance of the device and user subscription with the requested service	CheckEligibility	M	6.2	M M	6.5.1 6.5.2
Get the entitlement configuration document	AcquireConfiguration	M	6.2	M M	6.5.1 6.5.5
Get user consent	Display WebView	O	6.7	O	6.7
Manage user subscription.	ManageSubscription,	M	6.5.3	M	6.5.3
	Display WebView	O	6.7	O	6.7
	Subscription transfer (=ManageSubscription with `old_terminal_iccid`)	M O	6.5.3 8.3	M O	6.5.3 8.3
	Subscription transfer using a temporary token	O	6.5.7	O	6.5.7
	New eSIM subscription	M O	6.5.3 8.1	M O	6.5.3 8.1
Change the service status	ManageService	O	6.2	M M	6.5.1 6.5.4
Update the entitlement configuration from network	Push notification	C1	2.6	C1	2.6
	Polling	C1	7.3		

Table 150. Features & operations mapping for Primary ODSA

C1: IF *Push notification* IS NOT SUPPORTED THEN *POLLING* IS *M*

A.7 Data Plan and Data Boost Information feature

Procedure	Operation	Entitlement Client		SP Entitlement Server	
Authenticate the Subscriber Identity	Embedded EAP-AKA authentication	O	2.8.1	M	2.8.1
	Fast authentication	M	2.8.5	M	2.8.5
Get the entitlement configuration document	HTTP GET	M	9.1	M M	6.5.1 9.1
Get the real-time data boost configuration document	HTTP GET with boost_type	M	9.8	M	9.9 9.10
Get the user consent	Display webviews	O	9.11	O	9.11

Table 151. Features & operations mapping for Data Plan Information

A.8 Server Initiated ODSA feature

Procedure	Operation	Entitlement Client		SP Entitlement Server	
Authenticate the Subscriber Identity	Fast authentication	M	2.8.5	M	2.8.5
	Server to server authentication	M	2.8.3	M	2.8.3
Check the compliance of the device and user subscription with the requested service	CheckEligibility	M M	6.2 10.1	M M	6.5.2 10.1
Get the entitlement configuration document	AcquireConfiguration	M	6.5.6	M	6.5.6
Manage user subscription	ManageSubscription	M	6.5.3	M	6.5.3
Update the entitlement configuration from network	Push notification	C1	2.6	C1	2.6
	Polling	C1	7.3		

Table 152. Features & operations mapping for Server Initiated ODSA

C1: IF *Push notification* IS NOT SUPPORTED THEN *POLLING* IS *M*

A.9 Direct Carrier Billing Entitlement feature

Procedure	Operation	Entitlement Client		SP Entitlement Server	
Authenticate the Subscriber Identity	Embedded EAP-AKA authentication	O	2.8.1	O	2.8.1
	Fast authentication	M	2.8.5	M	2.8.5
	OAuth2.0/OIDC authentication	O	2.8.2	O	2.8.2
Get the entitlement configuration document	HTTP GET	M	11, 11.4.1	M	11, 11.4.1
Update the entitlement configuration from network	Push notification	O	11.4.1	O	11.4.1
Get user consent	Display WebView	O	11.6, 11.4.2	O	11.6, 11.4.2

Table 153. Features & operations mapping for Direct Carrier Billing

A.10 Private User Identity feature

Procedure	Operation	Entitlement Client		SP Entitlement Server	
Authenticate the Subscriber Identity	Embedded EAP-AKA authentication	M	2.8.1	M	2.8.1
	Fast authentication	M	2.8.5	M	2.8.5
Get the entitlement configuration document	HTTP GET or POST	M	12, 12.1, 12.2, 12.4	M	12, 12.1, 12.2, 12.4
Update the entitlement configuration from network	Push notification	O	12.2	O	12.2

Table 154. Features & operations mapping for Private User Identity

A.11 User and Device Information feature

Procedure	Operation	Entitlement Client		SP Entitlement Server	
Authenticate the Subscriber Identity	Embedded EAP-AKA authentication	O	2.8.1	M	2.8.1
	Fast authentication	M	2.8.5	M	2.8.5
	Use a temporary token for getPhoneNumber operation (AcquireTemporaryToken)	M M	6.1, 6.2 6.5.7	M M	6.2 6.5.7
Get the entitlement configuration document	GetPhoneNumber	M	13.1.1.1, 13.1.2, 6.5.8	M	13.1.1, 13.1.2, 6.5.8
	GetSubscriberInfo	M	13.2.1, 6.5.11	M	13.2.1, 6.5.11

Table 155. Features & operations mapping for User and Device Information

A.12 Device App authentication features

Procedure	Operation	Entitlement Client		SP Entitlement Server	
Authenticate the Subscriber Identity	Embedded EAP-AKA authentication	M	2.8.1	M	2.8.1
	Fast authentication	O	2.8.5	O	2.8.5
	Use a temporary token for AcquireOperatorToken operation (AcquireTemporaryToken)	M M	6.1, 6.2 6.5.7	M M	6.2 6.5.7 6.6.6
	TemporaryToken Error Handling	O	2.8.6	O	2.8.6
	AcquireOperatorToken	M	14.1.1 14.1.2	M	14.1.1 14.1.2
Consuming Operator Token	ValidateOperatorToken	M	14.1.6 14.1.6.1	M	14.1.6 14.1.6.1
	GetSubscriberDeviceInfo	M	14.1.6 14.1.6.2	M	14.1.6 14.1.6.2
	VerifyPhoneNumber	M	14.1.6 14.1.7	M	14.1.6 14.1.7
Consuming App token	Get3PAppInfo	M	14.2 14.2.1	M	14.2 14.2.1

Table 156. Features & operations mapping for App Authentication with OperatorToken

A.13 SatMode Entitlement feature

Procedure	Operation	Entitlement Client		SP Entitlement Server	
Authenticate the Subscriber Identity	Embedded EAP-AKA authentication	M	2.8.1	M	2.8.1
	Fast authentication	O	2.8.5	O	2.8.5
Get SatMode Entitlement configuration	HTTP GET	M	2.9 15.2	M	6.5.2 15.2
JS Callbacks for Webview	entitlementChanged() dismissFlow()	M	15.4	M	15.4

Table 157. Features & operations mapping for Phone Number Information

Annex B Document Management

B.1 Document History

Version	Date	Brief Description of Change	Approval Authority	Editor / Company
1.0	July 2018	First version	TG#11	J. Sicard / HPE
2.0	October 2018	Updated with changes detailed in CR1002	TSG	J. Sicard / HPE
3.0	Not Published	Added eSIM devices configuration and restructuring the document.	TSG	J. Sicard / HPE
4.0	December 2019	Adding Companion devices configuration. CR1003	TSG#38	J. Sicard / HPE
5.0	April 2020	Changed title, Chapter 6 reflects generic ODSA operations and parameters, Companion ODSA call flows are in separate Chapter 7, new Chapter 8 presents Primary ODSA call flows, clarifications added for POST content encoding and UserData response parameter, new logout callback function. CR1004 & CR1005	TSG39j	J. Sicard / HPE
6.0	January 2021	Chapter 6 reflects JS callback functions, poll and push mechanisms defined for delayed profile delivery in chapter 7, use case data plan information added in chapter 9, server initiated ODSA use case added in chapter 10, OID error processing, entitlement version handling, user agent format, obtaining identifiers with web sheet **CR1020** (CRs1006, 1007, 1008, 1011, 1012, 1013, 1014, 1015, 1016 & 1017)	TSG#42 ISAG#6	F. Schmitt / DT
6.1	April 2021	Adding CR numbers to V6.0 version history CR1024	TSG (email)	Paul Gosden / GSMA
7.0	Not Published	Changes the same as for V8.0		Paul Gosden / GSMA
8.0	January 2022	Added new use cases VoNR entitlement and primary ODSA via EAP-AKA. Enhanced companion & primary ODSA procedures, AppID handling, token management, error & eligibility handling. **CR1040** (CR1009, CR1018, CR1019, CR1021, CR1022, CR1023, CR1024, CR1025, CR1026, CR1027, CR1028,	TSG#46 ISAG#13	Florian Schmitt / DT

Version	Date	Brief Description of Change	Approval Authority	Editor / Company
		CR1029, CR1030, CR1031, CR1032, CR1033, CR1034, CR1043)		
9.0	December 2022	Added new use cases Direct Carrier Billing (Chapter 11) and Private User Identity (Chapter 12). Added temporary token & MSISDN retrieval functionality, enhanced auth mechanism, subscription transfer, discovery & JS callbacks. Added Annex B Feature Mapping. **CR1060** (CR1035, CR1042, CR1044, CR1045, CR1046, CR1047, CR1048, CR1049, CR1051, CR1052, CR1055, CR1056, CR1057)	TSG#50 ISAG#26	Florian Schmitt / DT
10.0	December 2023	**CR1080** (CR1061, CR1062, CR1063, CR1064, CR1065, CR1066, CR1067, CR1068, CR1069, CR1070, CR1071, CR1073, CR1074, CR1077)	TSG#54 ISAG#37	Kay Fritz / Vodafone
10.1	March 2024	**CR1088 v02** Non-Substantive Change.	TSG via email	Kay Fritz / Vodafone
11.0	April 2024	**CR1090** (CR1076, CR1078, CR1081, CR1082, CR1084, CR1085, CR1086, CR1087)	TSG#55 ISAG#40	Kay Fritz / Vodafone
12.0	February 2025	**CR1120** (CR1091, CR1092, CR1093, CR1094, CR1095, CR1096, CR1097, CR1098, CR1099, CR1100, CR1103, CR1104, CR1105)	TSG#58 ISAG#48	Kay Fritz / Vodafone
13.0	Jan 2026	**CR1160v8** (CR1121v2, CR1132v1, CR1133v1, CR1134v2, CR1135v2, CR1136v4, CR1137v1, CR1142v2, CR1161v3)	TSG#62 ISAG#58	Paul Gosden /GSMA

(*) Document History Version (in table above) is used as a reference for a configuration parameter in this document, so it requires to follow a specific pattern. This version number is expected to be defined as the following ABNF rule: 1*DIGIT”.”1*DIGIT. Some valid values could be: 6.0; 6.1; 10.0 or 11.10

B.2 Other Information

Type	Description
Document Owner	Terminal Steering Group (TSG)
Editor / Company	Kay Fritz / Vodafone

It is our intention to provide a quality product for your use. If you find any errors or omissions, please contact us with your comments. You may notify us at prd@gsma.com

Your comments or suggestions & questions are always welcome.